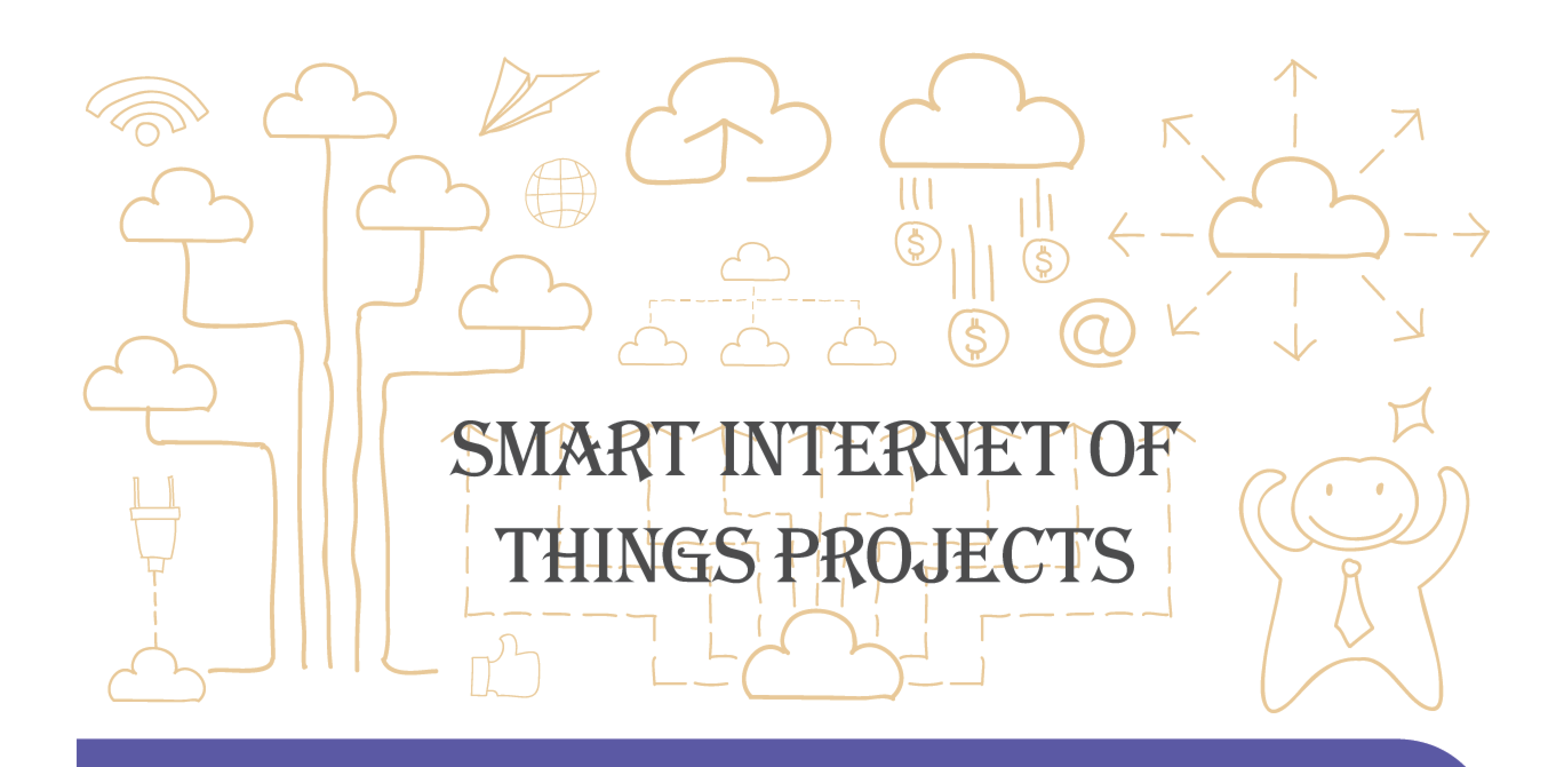


SMART INTERNET OF THINGS PROJECTS

智能物联网项目开发实战

[美] Agus Kurniawan 著　杜长营 译



清华大学出版社

智能物联网项目开发实战

[美] Agus Kurniawan　著

杜长营　译

清华大学出版社

北　京

内 容 简 介

本书详细阐述了智能物联网开发的实现过程，主要包括决策系统、机器视觉系统、自动机器车、语音技术和数据云等内容。此外，本书还提供了相应的示例、代码，以帮助读者进一步理解相关方案的实现过程。

本书适合作为高等院校计算机及相关专业的教材和教学参考书，也可作为相关开发人员的自学教材和参考手册。

北京市版权局著作权合同登记号 图字：01-2017-7946

图书在版编目（CIP）数据

智能物联网项目开发实战/（美）阿古斯•库尼亚万（Agus Kurniawan）著；杜长营译. —北京：清华大学出版社，2018
（书名原文：Smart Internet of Things Projects）
ISBN 978-7-302-49221-4

Ⅰ. ①智… Ⅱ. ①阿… ②杜… Ⅲ. ①互联网络-应用 ②智能技术-应用 Ⅳ. ①TP393.4 ②TP18

中国版本图书馆 CIP 数据核字（2017）第 331864 号

责任编辑： 贾小红
封面设计： 刘　超
版式设计： 魏　远
责任校对： 马子杰
责任印制： 刘海龙

出版发行： 清华大学出版社
网　　址： http://www.tup.com.cn，http://www.wqbook.com
地　　址： 北京清华大学学研大厦 A 座　　**邮　　编：** 100084
社 总 机： 010-62770175　　**邮　　购：** 010-62786544
投稿与读者服务： 010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈： 010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者： 北京密云胶印厂
经　　销： 全国新华书店
开　　本： 185mm×230mm　　**印　　张：** 13.5　　**字　　数：** 270 千字
版　　次： 2018 年 1 月第 1 版　　**印　　次：** 2018 年 1 月第 1 次印刷
印　　数： 1～3000
定　　价： 69.00 元

产品编号：075238-01

智能物联网项目

一起探索如何开发你的智能物联网项目，并为你的世界带来全新的连接。

译 者 序

世界是普遍联系的，科技的发展让这种联系变得更加便捷和迅速，从而大大提升了人们的生活体验。

在互联网诞生之后，它迅速地凝聚了海量的信息，这些信息源自人类社会却又远远超出任何一个人类个体的认知范围。人们可以根据从互联网获得的有效信息来进行决策，优化自己应对世界的运作模式。但大千世界除了人类之外，还有数之不尽的物体，它们同样是信息的载体，是尚未被充分利用起来的一部分世界。科技在于探索未知，在于创造可能。我们不禁思索：这些物体是不是也可以通过网络进行连接呢？如果连接被启动，网络得以构建，则万物“苏醒”，物物相息，它们将跨越时空的鸿沟，更加灵活自主地运转，更加高效地优化配置——这是多么振奋人心的未来！也许到了这一天，我们一直期待的人工智能才真正得以实现。

在翻译这本实战类书籍的过程中，我们也体会到智能物联网涉及的技术领域包含大量内容：诸如计算机科学、通信工程、人工智能、软件工程、数学等学科的知识被融为一体。显然，完成这项事业，需要各专业领域的人才通力合作，攻克一系列史无前例的技术难题。我们也可以看到，在世界各主要国家的政策支持下，在国内外各研究机构的潜心钻研下，在大型跨国企业集团的积极参与下，这些难题正在逐渐被破解。

EPOSS 在 Internet of Things in 2020 报告中分析预测，物联网的发展将经历四个阶段：2010 年之前 RFID 被广泛应用于物流、零售和制药领域；2010—2015 年物体互联；2015—2020 年物体进入半智能化；2020 年之后物体进入全智能化。

2018 年以后的世界会是什么样的呢？我们期待并为之努力着。

最后，苏辉、杜长德、栾思焘、王喆、郑聪、陈香宇、沈旻、王烈征、张博、张弢、张颖、李垚、李强、李秋霞、黄丽臣也参与了本书的翻译工作，他们花费了很多精力并提出了不少宝贵意见；感谢我的编辑徐瑞鸿老师在翻译过程中与我反复沟通以提升文字质量。谨以此向他们表达我衷心的谢意。

杜长营

前　　言

物联网（IoT）是指连接各种物理设备到网络并能控制它们的突破性技术。创建基本的物联网项目是很普通的，但是设想一下如果一个智能物联网项目能够从物理设备抽取出数据，它将能够实现自我决策。

智能物联网项目是实施物联网和智能系统结合方案的重要参考。基本的统计知识和各种统计科学、机器学习的算法已经被用来加速实现在物理设备上集成控制系统。本书包含一些物联网项目，如制作一个智能温度控制器，制作你自己的机器视觉项目，制作一个自动控制的移动小车，通过语音命令控制物联网项目，以及利用云技术、数据科学来帮助创建物联网项目。

希望本书能对你有帮助，让你的技能提升一个台阶。

本书包括

第 1 章，让物联网项目变得智能，帮助用户了解一些物联网设备，如 Arduino 和 Raspberry Pi（树莓派）。介绍一些统计和数据科学的 Python 库，了解它们的作用。

第 2 章，将决策系统用于物联网工程，帮助用户了解如何在物联网设备上构建控制系统。包括了解一些与控制系统相关的 Python 库，学习如何在 IoT 板上实现决策系统。

第 3 章，搭建机器视觉，探索如何通过摄像机让机器能够“看到”事物，并在训练机器检测和跟踪物体时对机器视觉有所理解。另外，本章也会介绍一些摄像机模块方面的知识。

第 4 章，制作自动机器车，探索如何制作机器车。通过集成一些传感器和驱动器让小车自行运动而不需要人为的干预。学习如何导航，同时也可以在计算机上控制它。

第 5 章，在物联网项目中添加语音技术，使 IoT 板“说话”。了解各种声音和语音模块。

第 6 章，为物联网项目搭建数据云，探索如何为物联网项目应用云平台。物联网项目的后端基础建设也是很重要的。当在不同地理位置处获取传感器数据时更需要注意。

你需要准备什么

你需要拥有 Raspberry Pi、Arduino 和一些本书中需要用到的电子组件。

适合的读者

本书适合希望学习如何将各种机器学习算法集成在物联网项目里的读者。你会学习到如何在真实的物联网项目里实现机器学习功能。但是你不需要对 Raspberry Pi 和 Arduino 有任何经验。

格式约定

在本书中，你会发现一些文字格式有所区别。这里给出一些例子说明它们的意义。

文字中的代码、数据库表名、文件夹名字、文件名、文件扩展、路径、链接、用户输入和 Twitter handles 都如下所示："我们用 sm.OLS()实现线性回归"。

块状代码如下：

```
import RPi.GPIO as GPIO
import time

led_pin = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(led_pin, GPIO.OUT)
```

如果想让读者着重注意某部分代码，则将其设为加粗：

```
try:
    while 1:
        print("turn on led")
        GPIO.output(led_pin, GPIO.HIGH)
        time.sleep(2)
        print("turn off led")
        GPIO.output(led_pin, GPIO.LOW)
        time.sleep(2)

except KeyboardInterrupt:
    GPIO.output(led_pin, GPIO.LOW)
    GPIO.cleanup()

print("done")
```

命令行的输入和输出格式如下：

```
$ mkdirgps_web
```

```
$ cdgps_web
$ nano gspapp.py
```

注意：表示警告或者重要的说明。

提示：表示提示和技巧。

读者反馈

我们欢迎读者反馈，让我们了解读者对于本书的看法——喜欢的和不喜欢的部分。读者反馈对我们非常重要，因为它能帮助我们了解读者真正学到的部分。

读者可以通过发送邮件到 feedback@packtpub.com，只需要在主题里写上书的题目即可。

如果读者对一个主题有专业的看法并且想要对写作或者对书做一些贡献，可参考我们的作者指南：www.packtpub.com/authors。

消费者支持

现在你是书的拥有者，我们有一些内容提供给你，帮助你最大化你购买的价值。

下载示例代码

你可以用你的账户在 http://www.packtpub.com 下载到示例代码。如果你在其他地方购买本书，可以访问 http://www.packtpub.com/support 并注册，我们把文件直接发给你。

你可以通过以下步骤下载到代码：

（1）用邮箱地址和密码登录或者注册我们的网站。

（2）将鼠标指针移动到顶部的 SUPPORT 一栏。

（3）单击 Code Downloads & Errata。

（4）在 Search 框中输入书的名字。

（5）选择要下载代码的书。

（6）选择购买书的位置的下拉菜单。

（7）单击 Code Download。

你也可以在 Packt 出版社网站的这本书的主页单击 Code Files 按钮下载，还可以通过在 Search 框中搜索这本书的名字找到。请注意你需要登录你的 Packt 账户。

文件下载后，请确认你的解压软件是最新版本：

- WinRAR / 7-Zip for Windows

- Zipeg / iZip / UnRarX for Mac
- 7-Zip / PeaZip for Linux

本书的代码也被托管在 GitHub，网址为 https://github.com/PacktPublishing/Smart-Internet-of-Things-Projects。我们也把其他书籍的代码和视频放在 https://github.com/PacktPublishing/。

勘误

虽然我们已经努力确保内容正确，但是错误仍难避免。如果你发现文字或者代码的错误并能告知我们，我们将非常感激。这样可消除其他读者的困惑，也能帮助我们提高后面版本的质量。你可以访问 http://www.packtpub.com/submit-errata，选择你要勘误的书，单击 Errata Submission Form 并输入勘误的细节。一旦你的勘误得到验证，我们会接收你提交的信息并将勘误上传到我们的网站，或者添加到勘误章节的列表里。

访问 https://www.packtpub.com/books/content/support 并在搜索框中输入书的名字，在 Errata 一节中可以看到之前的勘误。

盗版行为

盗版行为在互联网上非常常见。在 Packt，我们非常严肃地保护我们的版权。如果你看到任意形式的非法复制，请立即提供给我们网站地址和名字，我们将追究赔偿。

请把有盗版嫌疑的材料发送到 copyright@packtpub.com。

对于你对我们作者的保护，我们不胜感激，我们将给您提供有价值内容的权益。

问题

如果你对本书有任何问题，欢迎联系 questions@packtpub.com，我们将竭尽全力解决你的问题。

关 于 作 者

Agus Kurniawan 是一位讲师，也是 IT 顾问和作家。他在各种软硬件开发，提供培训和研讨会材料还有技术写作等方面有 14 年的经验，连续 12 年获得微软最有价值专家（MVP）奖。

现在他在印度尼西亚大学的计算科学系和印度尼西亚三星研究院做研究工作并从事网络和安全系统的教学工作。目前，他正在德国柏林自由大学攻读计算科学的博士学位。他的个人博客是 http://blog.aguskurniawan.net，个人 Twitter 是@agusk2010。

关于审稿人

Phodal Huang 在硬件开发和 Web 开发方面有超过 6 年的经验。他毕业于西安文理大学，目前在 ThoughtWorks 公司做顾问，是迷你物联网项目（https://github.com/phodal/iot）和电子书《设计 IoT》（http://designiot.phodal.com 仅中文版）的作者。同时他还是 *Learning Internet of Things* 的审稿人，截至成书时，他自己的书 *Design Internet of Things* 也正在出版中。

Phodal Huang 热爱设计，写作，你可以在他的个人网站 http://www.phodal.com 找到更多他的信息。他也热衷于开源软件的开发，你可以在 http://github.com/phodal 找到他。

目　　录

第 1 章　让物联网项目变得智能 1
统计学和数据科学简介 1
用于统计计算和数据科学的 Python 3
用于统计计算和数据科学的 Python 库 5
编写一个用于统计的简单程序 6
物联网设备和平台 8
Arduino 8
Raspberry Pi 10
BeagleBone Black and Green 12
基于 ESP8266 MCU 的物联网开发板 13
基于 TI CC3200 MCU 的物联网开发板 15
物联网设备感知和启动 17
Arduino 设备感知和启动 17
Raspberry Pi 设备感知和启动 25
为房间建造一个智能温度控制器 31
PID 控制器介绍 31
用 Python 实现 PID 控制器 32
使用 PID 控制器控制房间温度 40
总结 44
引用 44

第 2 章　将决策系统用于物联网工程 45
决策系统和机器学习基本介绍 45
用于决策系统的贝叶斯 45
用于决策系统的模糊逻辑 46
搭建决策系统所需的 Python 函数库 48
贝叶斯模型 48
模糊逻辑 53

搭建一个简单的基于贝叶斯理论的决策系统 55
将决策系统和物联网项目结合 58
搭建基于决策系统的物联网 60
布线 60
编写 Python 程序 61
测试 66
提高 67
总结 68
引用 68

第 3 章　搭建机器视觉 69
机器视觉的基本介绍 69
OpenCV 函数库介绍 70
在 Raspberry Pi 上配置 OpenCV 70
使用 OpenCV 编写一个简单的程序 75
使用摄像机模块 78
基于 CSI 接口的摄像机模块 78
基于 USB 接口的摄像机模块 80
基于串行（serial）接口的摄像机模块 80
多种接口的摄像机模块 81
从 OpenCV 函数库访问摄像机模块 82
介绍用于机器视觉的模式识别 84
为移动的物体搭建视觉识别系统 86
搭建 IoT 机器视觉 88
在 Raspberry Pi 上部署 Pixy CMUcam5 88
装配 89
升级 Pixy CMUcam5 固件 89
测试 89
总结 94
引用 95

第 4 章　制作自动机器车 97
自动系统介绍 97
介绍移动机器人 99

搭建机器车 100
　DIY 机器人平台 100
　集成的机器人平台 102
使用 Pololu Zumo robot for Arduino 104
用计算机控制机器车 109
使用 GPS 模块导航 117
介绍地图引擎平台 124
制作基于 GPS 的小车 128
制作自动机器车 130
总结 131
引用 131

第 5 章　在物联网项目中添加语音技术 133
语音技术介绍 133
声音传感器和驱动器介绍 134
语音技术的模式识别介绍 143
介绍语音和声音模块 143
为物联网项目增加语音控制 145
　设置 EasyVR shield 3 145
　创建语音命令 148
　给语音板布线 151
　编写 Sketch 程序 151
　测试 157
让 IoT 板说话 157
　设置 157
　布线 157
　编写 Sketch 程序 158
　测试 159
让 Raspberry Pi 说话 159
　设置 159
　编写 Python 程序 162
　下一步是什么？ 163
总结 163

引用 163
第 6 章　为物联网项目搭建数据云 165
对云技术的介绍 165
介绍基于云的数据科学 166
连接 IoT 板到云服务器 167
微软 Azure IoT 167
亚马逊 AWS IoT 168
Arduino 云 168
使用微软 Azure IoT Hub 180
设置微软 Azure IoT Hub 180
注册 IoT 设备 182
编写程序 186
构建科学型云平台 192
部署 Azure 机器学习 193
发布到 Azure ML 作为 Web 服务 194
构建带有科学型数据云的 IoT 应用 196
总结 196
引用 197

第 1 章　让物联网项目变得智能

我们将从回顾基本的统计知识开始，然后学习如何在 Arduino 和 Raspberry Pi 等物联网设备（IoT）上进行检测和执行。我们还将会介绍各种与统计和数据有关的 Python 函数库。这些函数库对开发贯穿本书的项目很有帮助。

本章将会学到如下知识点：

- 几种统计和数据分析方法的介绍
- 几种物联网设备和平台的回顾
- 通过外部设备在物联网设备上检测和执行
- 开发一个智能物联网项目

现在让我们开始吧！

统计学和数据科学简介

假设你想要知道你房间的温度，所以在一天当中，你用某种特定的工具每隔一小时测量一下。这些数据是必需的，因为你想要据此决定是否需要买一个空调。当测量结束之后，你获得了一列温度数据。测量的结果如表 1-1 所示。

表 1-1

时　间	温度（摄氏度）	时　间	温度（摄氏度）
01:00	18	13:00	28
02:00	17	14:00	29
03:00	18	15:00	28
04:00	19	16:00	27
05:00	20	17:00	25
06:00	20	18:00	24
07:00	21	19:00	24
08:00	22	20:00	23
09:00	22	21:00	22
10:00	24	22:00	20
11:00	25	23:00	19
12:00	26	24:00	19

表 1-1 显示了温度数据。对于这种情况，你需要一些统计学知识。一些统计学名词，如平均数（mean）、中位数（median）、方差（variance）和标准差（standard deviation）需要提前熟知。

假设有一个样本，样本中有 n 个数据，分别用 $x1, x2, x3,\cdots, xn$ 来指代。可以用下列公式来计算平均数、中位数、方差和标准差。

$$mean = \overline{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$median = value\ on\ position \frac{n+1}{2}, if\ n\ is\ odd$$

$$median = value\ on\ position\ between\ n/2\ and\ (n/2)+1, if\ n\ is\ even$$

$$variance = s^2 = \frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n-1}$$

$$standard\ deviation = s = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n-1}}$$

提示：为了计算中位数，应该将数据按升序排列。

用表 1-1 中的数据，可以据如上公式计算数据的平均数、中位数、方差和标准差。这些值应该分别是 22.5、22、12.348 和 3.514。

为了理解数据的模式，可尝试着用图表形式来显示，如用 Microsoft Excel。结果如图 1-1 所示。

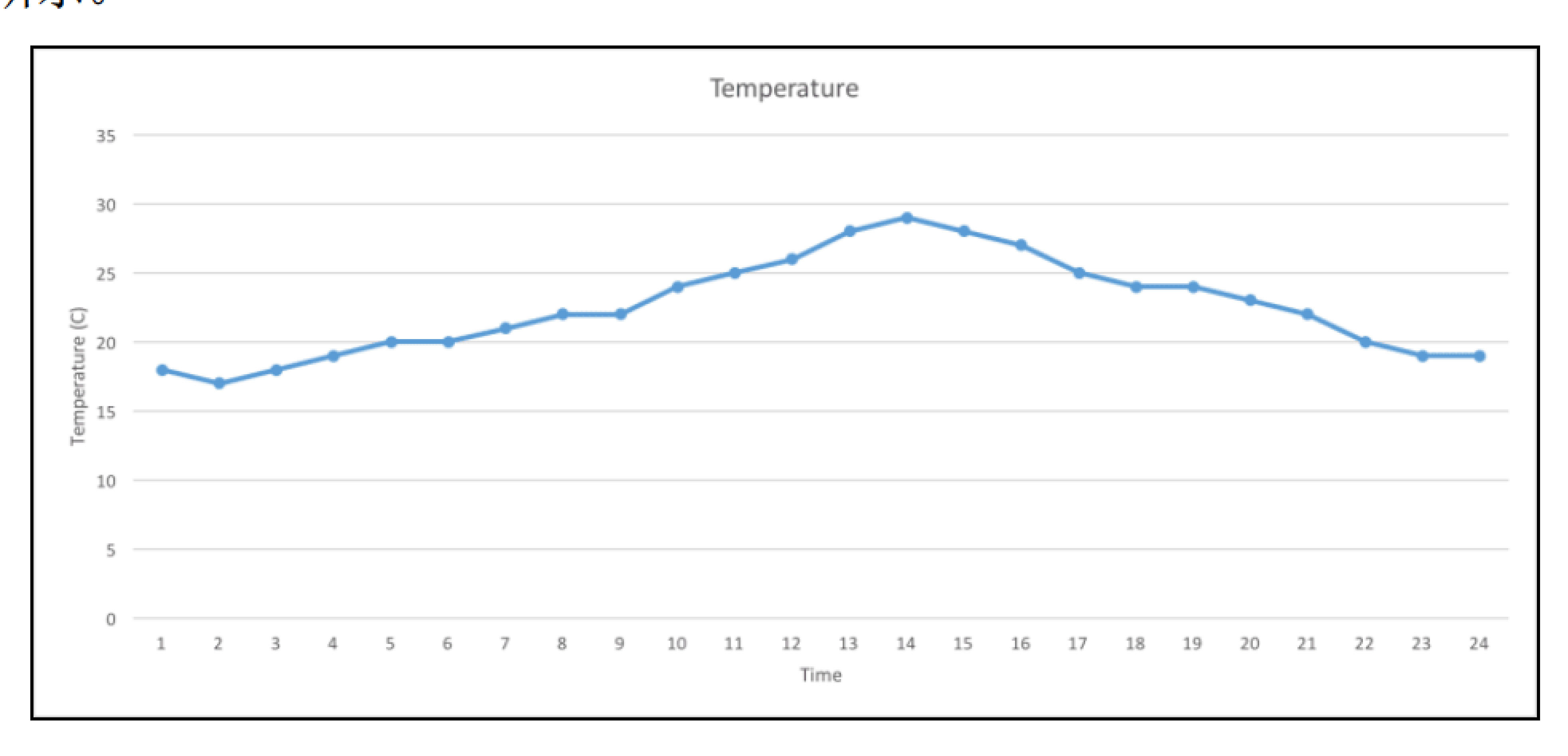


图 1-1

你会看到房间的平均温度是 22.5 摄氏度。温度的最大值和最小值分别是 29 和 17。根据这些信息，你可以考虑想要买哪种类型的空调。

另外，可以拓展一下你的工作，连续一周测量房间内的温度。测量之后，你可以把结果用图表形式反映出来，例如，还是用 Microsoft Excel。图 1-2 显示了温度测量图表的一个例子。

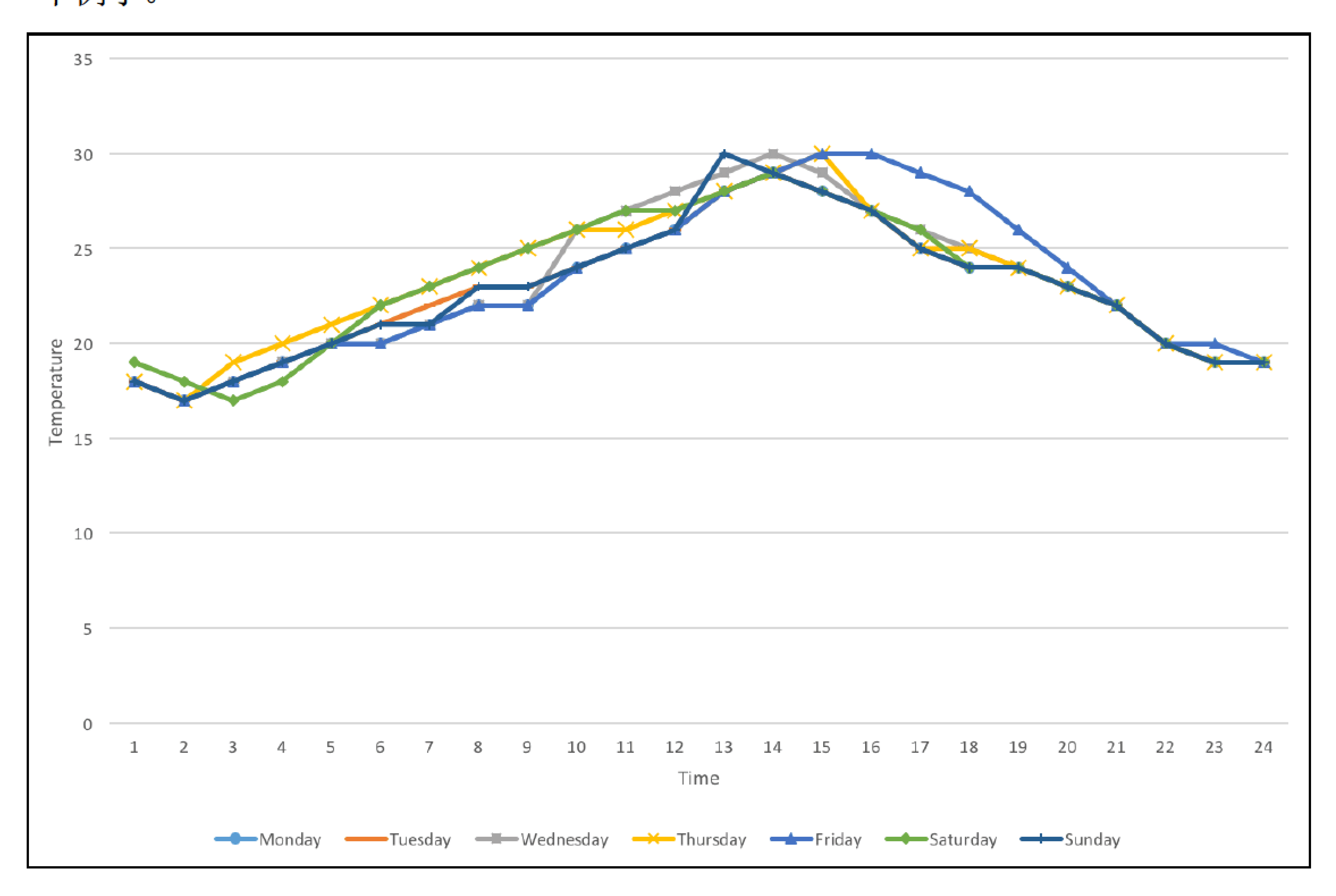


图 1-2

图 1-2 显示了房间温度每天都在变化。如果持续一年每天坚持测量，应该可以清楚地看到房间温度的变化趋势。数据科学的知识可以提高从数据中学习的能力。当然，有些统计计算和机器学习的知识也会被包含进来，帮助用户更好地理解数据的模式。

这本书将会帮助读者了解如何把数据科学和机器学习应用到真实案例中。本书侧重点将会在物联网领域。

用于统计计算和数据科学的 Python

Python 是一种被广泛使用的通用编程语言。入门编程，Python 是一个很好的选择。

Python 提供了简单的编程语法和数量繁多的函数接口，可以用这些函数接口来拓展程序。

为了能在计算机上使用 Python，可以从 https://www.python.org/downloads/下载和安装（假定操作系统没有默认安装）。安装完成之后，可以通过在终端或者 Windows 系统中的命令行里输入下列命令来运行 Python 程序：

```
$ python
```

提示： *忽略$标志，只需在终端里输入python，适用于Python 2.x。*

运行该命令后，应该可以看到 Python 命令行界面，如图 1-3 所示。

```
agusk — python — 80×23
agusk$ python
Python 2.7.10 (default, Oct 23 2015, 19:19:21)
[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.0.59.5)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

图 1-3

如果安装了 Python 3，通常会通过如下命令行运行程序：

```
$ python3
```

终端里看到的 Python 3 界面，如图 1-4 所示。

```
agusk — Python — 80×23
agusk$ python3
Python 3.5.1 (default, Jan 22 2016, 08:54:32)
[GCC 4.2.1 Compatible Apple LLVM 7.0.2 (clang-700.1.81)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

图 1-4

接下来是什么？

有很多 Python 资料可以帮助学习如何用 Python 编写程序。推荐通过 https://www.python.org/doc/阅读 Python 文档。也可以阅读 Python 相关的书籍来加速学习进程。这本书将不会涉及基本 Python 编程方面的主题。

用于统计计算和数据科学的 Python 库

Python 拥有庞大的社区群体，帮助社区里的成员快速地学习和分享。很多社区已经开源了统计计算和数据科学的相关函数库，这些函数库可以用在工作中。接下来会使用这些函数库来实现相关代码。

以下是一些用于统计计算和数据科学的 Python 函数库。

NumPy

NumPy 是一个为了高效使用 Python 进行科学运算的基本函数库，可以处理 N 维数组并且嵌入 C/C++和 Fortran 代码，也提供了可用于线性代数、傅里叶变换和生成随机数的功能。

NumPy 官方网站可通过网址 http://www.numpy.org 进行访问。

Pandas

Pandas 是一个用于解决类似表格结构的函数库，这种结构被称为 DataFrame 对象。它像 NumPy 中的数组对象一样，也拥有强大的能力，且支持高效的数学运算。

关于 Pandas 的更多信息，可以通过网址 http://pandas.pydata.org 进行访问。

SciPy

SciPy 是 NumPy 函数库的一个拓展，包含了可用于线性代数、插值、集合、聚类等的很多功能。

SciPy 的官方网站可通过网址 http://scipy.org/scipylib/index.html 进行访问。

Scikit-learn

Scikit-learn 是 Python 中最流行的用于机器学习的函数库，提供了很多有用功能，如数据预处理、分类、回归、聚类、降维和模型选择。

更多关于 Scikit-learn 的信息可以通过网址 http://scikit-learn.org/stable/进行访问。

Shogun

Shogun 是一个 Python 的机器学习函数库，侧重于大规模核方法的运算，如支持向量机（SVM）函数库中包含了大量不同的 SVM 实现方式。

官方网站可以通过网址 http://www.shogun-toolbox.org 进行访问。

SymPy

SymPy 是一个支持符号数学计算的 Python 函数库，可以支持微积分、代数、集合、离散数学、量子物理及许多其他运算。

官方网站可以通过网址 http://www.sympygamma.com 进行访问。

Statsmodels

Statsmodels 是一个可以用来处理数据、预估统计模型和测试数据的 Python 模块。

可以通过官方网站 http://statsmodels.sourceforge.net 来获取更多关于 Statsmodels 的信息。

编写一个用于统计的简单程序

在"统计学和数据科学简介"一节中已经测量了房间的温度，现在将试着用 Statsmodels 进行一些简单的统计计算。下面将使用测量结果数据，并且为此数据实现一个简单的线性回归模型。

首先，应该安装 Statsmodels。该函数库依赖一些其他函数库，如 NumPy、SciPy、Pandas 和 patsy。可以使用 pip 安装。输入下列命令：

```
$ pip install numpy scipy pandas patsy statsmodels
```

如果遇到关于安全权限的问题，可以使用 sudo 来运行命令：

```
$ sudo pip install numpy scipy pandas patsy statsmodels
```

如果计算机中没有安装 pip，可以按照 https://pip.pypa.io/en/stable/installing/中的指示来安装。

创建一个 Python 程序用于测试。如下列脚本所示：

```
import numpy as np
import statsmodels.api as sm
```

```
# 房间温度
Y = [18, 17, 18, 19, 20, 20, 21, 22, 22, 24, 25, 26, 28, 29, 28, 27,
25, 24, 24, 23, 22, 20, 19, 19]
X = range(1, 25)
X = sm.add_constant(X)

model = sm.OLS(Y, X)
results = model.fit()

# 打印
print(results.params)
print(results.tvalues)


print(results.t_test([1, 0]))
print(results.f_test(np.identity(2)))
```

使用 sm.OLS()来实现一个线性回归算法。接着，用 model.fit()来实现预测。最后，打印出计算结果。将该程序保存为 ch01_linear.py 的脚本文件。

现在，可以通过下列命令来运行该程序：

```
$ python ch01_linear.py
```

如果已经安装了 Python 3，可以通过下列命令来运行该程序：

```
$ python3 ch01_linear.py
```

可以在图 1-5 中看到该程序运行结果。该截图中显示的是使用 Python3 的运行结果。

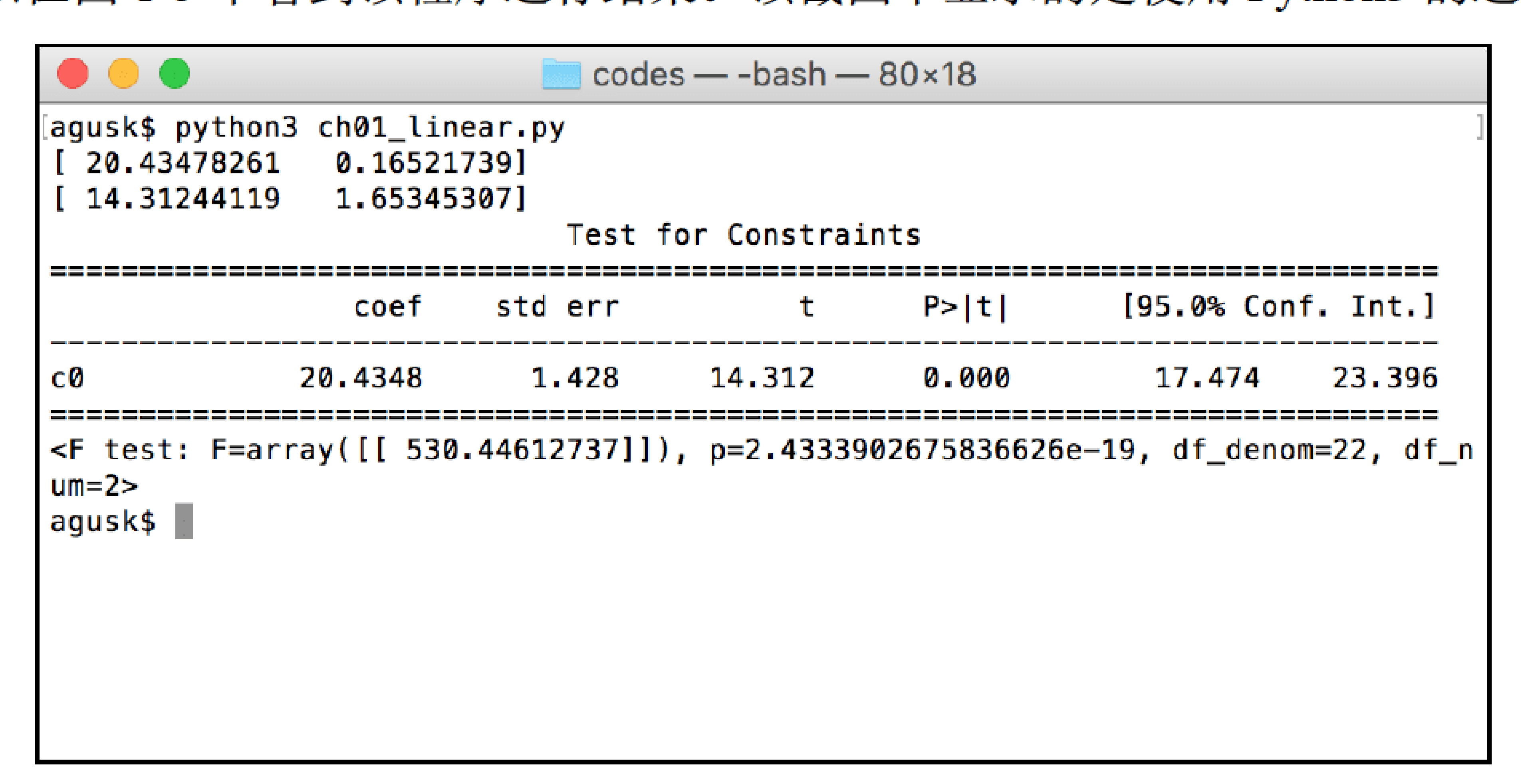


图 1-5

物联网设备和平台

物联网平台拥有可连接到互联网并且和其他平台交互的能力。总的来说，从设备平台的角度来谈论物联网是个比较大的话题。本节将会探索几种在客户端被广泛使用的物联网设备。

Arduino

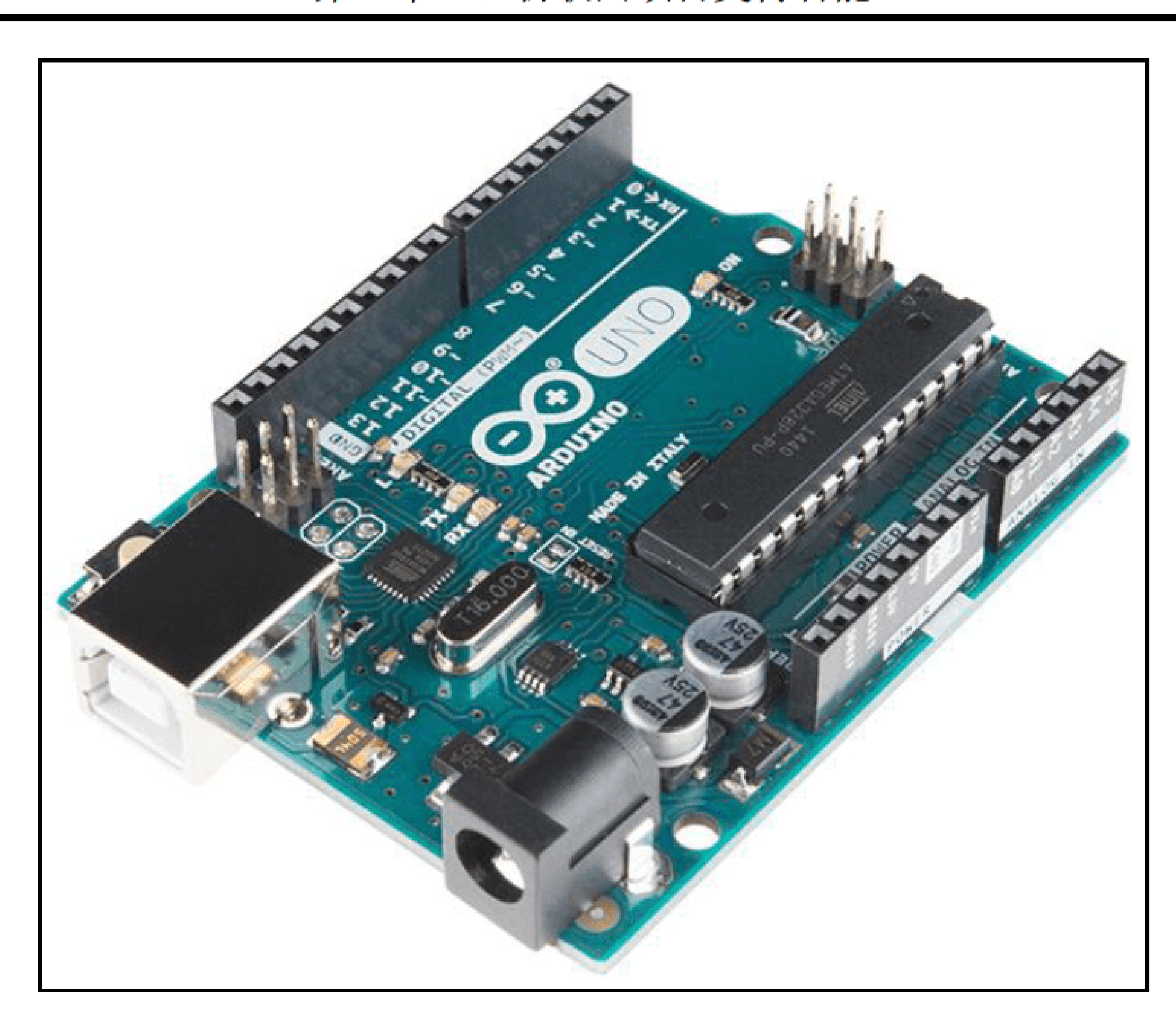
Arduino 是一个被广泛使用的开发板。该开发板在嵌入式社区中广为人知。大多数 Arduino 通过 Atmel AVR 编写，但是一些开发板使用和 Arduino 联合开发的其他 MCU。现在，Arduino 开发板通过 Arduino.cc 和 Arduino.org 编写。其他公司也制造开发板，这些开发板都和 Arduino 兼容。这是因为 Arduino 的创始人已经公开了开发板体系，这样每个人都可以编写自己的 Arduino。使用中请确保所使用的开发板和软件是属于同一家公司的产品。

为了拓展 Arduino 的 I/O 和功能，可以使用 Arduino shields。有很多不同的 Arduino shields，每个都有不同的功能，如蓝牙、Wi-Fi、GSM、温度、湿度检测器。使用 Arduino shield 的便利在于它允许用户集中注意力于开发板的开发过程。只需要把 Arduino shield 贴在 Arduino 开发板上，不需要任何焊接。

我们将从 Arduino.cc 上回顾一些 Arduino 开发板。可以通过访问网址 http://www.arduino.cc/en/Products/Compare 来阅读来自 Arduino.cc 的所有开发板的详细对比。我们将会回顾诸如 Arduino Uno、Arduino 101 和 Arduino MKR1000 这样的 Arduino 开发板。

Arduino Uno 模块被广泛用于 Arduino 开发。它基于 MCU ATmega328P 微型控制器编写。该开发板提供了几种电子和模拟 I/O 脚针，可以将感知器和执行设备贴到这上面。SPI 和 I2C 协议也被 Arduino Uno 提供。如果想了解更多关于该开发板的信息，推荐在 http://www.arduino.cc/en/Main/ArduinoBoardUno 阅读开发板的详细资料。Arduino Uno 开发板的样子如图 1-6 所示。

Arduino 101 在 I/O 脚针方面和 Arduino Uno 相同。Arduino 101 运行 Intel Curie，http://www.intel.com/content/www/us/en/wearables/wearable-soc.html 作为其核心模块。该开发板有内置的蓝牙模块。如果想使 Arduino 101 设备连接到一个 Wi-Fi 网络，应该添加一个额外的 Wi-Fi shield。推荐使用 Arduino Wi-Fi Shield 101，网址为 http://www.arduino.cc/en/Main/ArduinoWiFiShield101。图 1-7 所示为 Arduino 101 开发板。



来源：https://www.sparkfun.com/products/11021

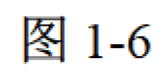
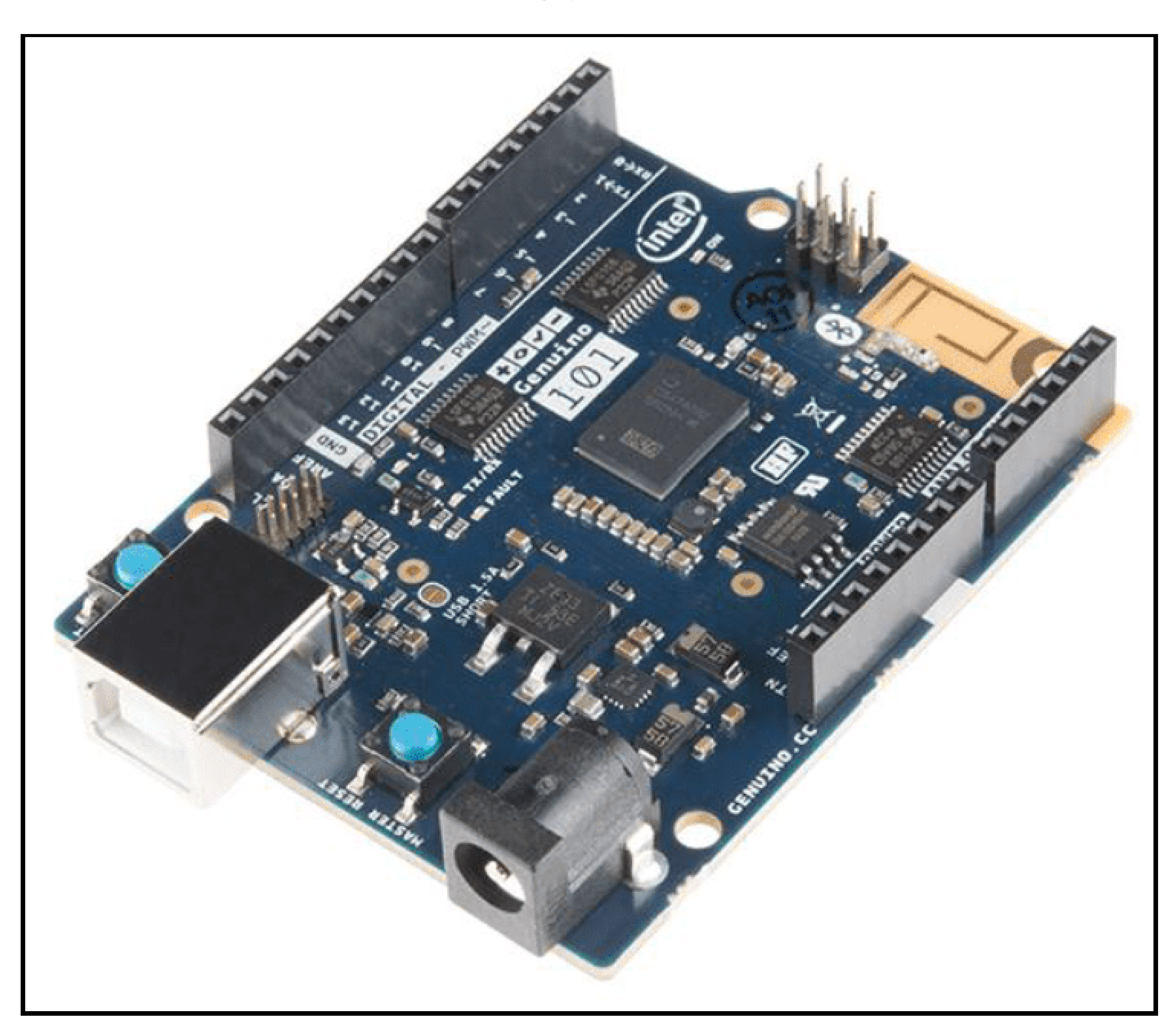


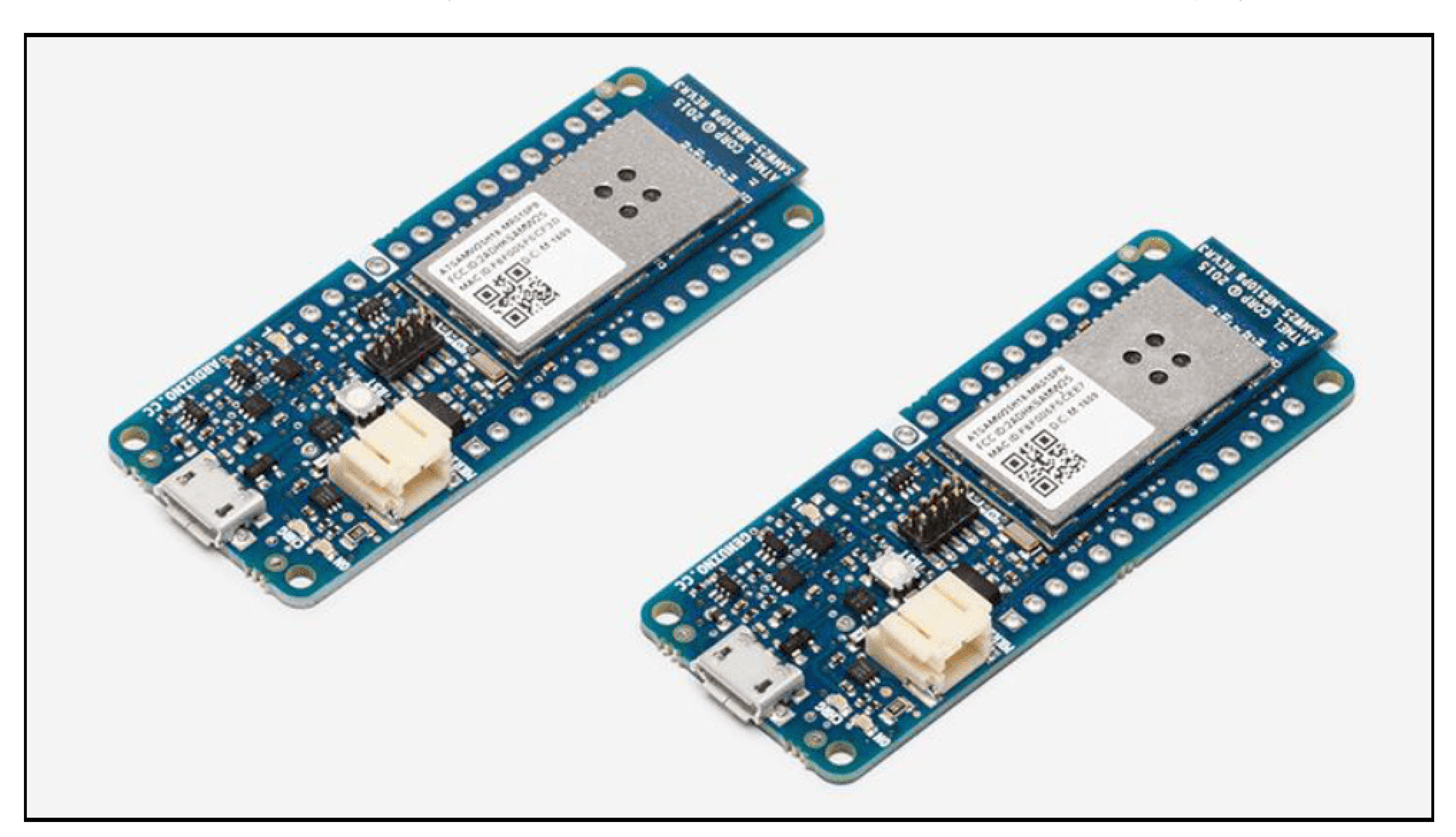
图 1-6



来源：https://www.sparkfun.com/products/13850

图 1-7

Arduino MKR1000 在该书编写时是一种新型开发板。此开发板使用 Atmel ATSAMW25 系统芯片，该芯片提供了内置 Wi-Fi 模块。推荐使用该开发板作为针对 Arduino 平台的物联网解决方案，因为 Wi-Fi 模块、WINC1500 被 SSL 和 ECC508 CryptoAuthentication 支持。更多关于该开发板的信息可以通过网址 http://www.arduino.cc/en/Main/ArduinoMKR1000 查看。图 1-8 所示为 Arduino MKR1000 开发板。



来源：http://www.arduino.cc/en/Main/ArduinoMKR1000

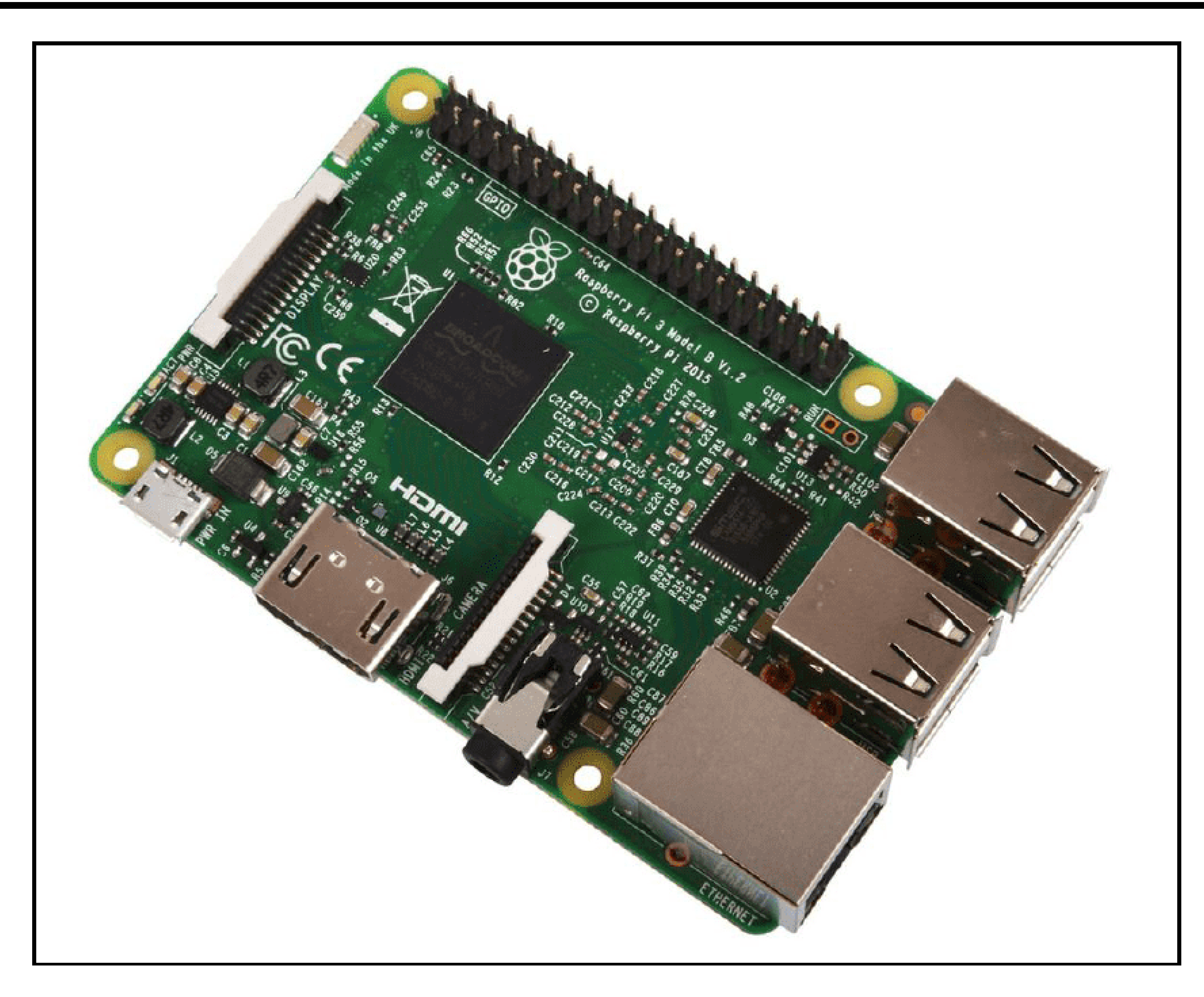
图 1-8

Raspberry Pi

Raspberry Pi 是 Eben Upton 所开发的低成本信用卡大小的计算机。它是为了教育目的研发的微型计算机。为了熟悉 Raspberry Pi 的所有模块，可以访问网址 https://www.raspberrypi.org/products/。下面列出 Raspberry Pi 3 Model B 和 Raspberry Pi Zero 的详细解释。

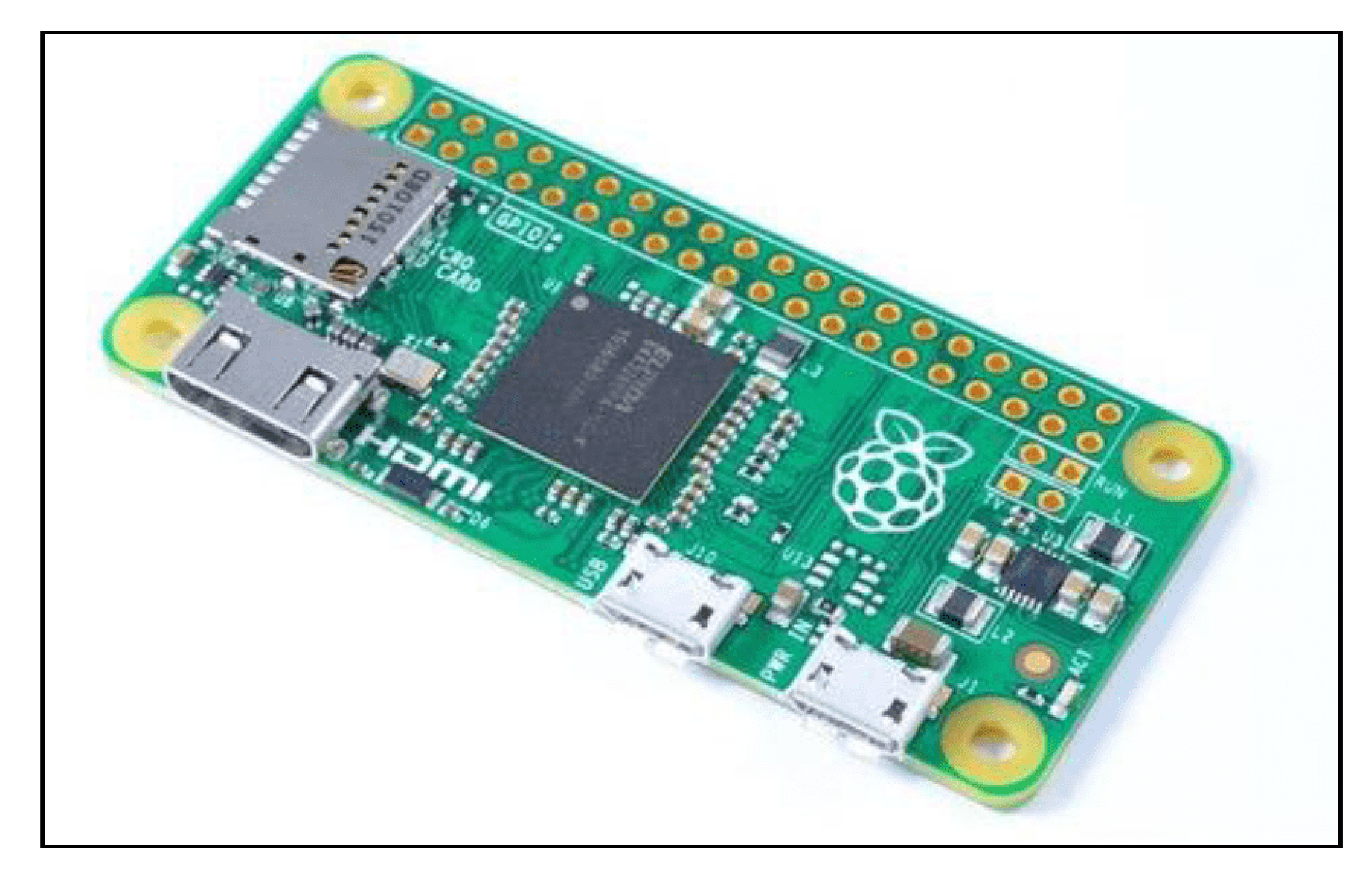
Raspberry Pi 3 Model B 是 Raspberry Pi 的第三代。该开发板包含了一块 Quad-Core 64-位 CPU、Wi-Fi 和蓝牙。强烈推荐用 Raspberry Pi 3 Model B 作为物联网解决方案。图 1-9 所示为 Raspberry Pi 3 Model B 开发板。

Raspberry Pi Zero 是一块小型计算机，大概有 Model A+的一半大。它运行一块单核 CPU，没有网络模块，但是它提供一个可连接至显示器的微型 HDMI 接口。由于网络模块缺失，故需要一个额外模块，如 Ethernet USB 或者 Wi-Fi USB 来将 Raspberry Pi Zero 连接至网络。图 1-10 所示为 Raspberry Pi Zero 开发板。



来源：https://thepihut.com/collections/raspberry-pi/products/raspberry-pi-3-model-b

图 1-9



来源：https://thepihut.com/collections/raspberry-pi-zero/products/raspberry-pi-zero

图 1-10

BeagleBone Black and Green

BeagleBone Black（BBB）Rev C 是一个基于 AM335x 处理器的开发包，集成了一块运行速度最大达到 1GHz 的 ARM Cortex™-A8 处理器。BBB 比 Raspberry Pi 性能更高。BBB 开发板也提供集成在开发板内闪存中的内部 4GB 8-bit eMMC。

BBB 支持各种操作系统（OS），如 Debian、Android 和 Ubuntu。想要了解更多关于 BBB 的信息，请访问 https://beagleboard.org/black。

图 1-11 所示为 BeagleBone Black 开发板。



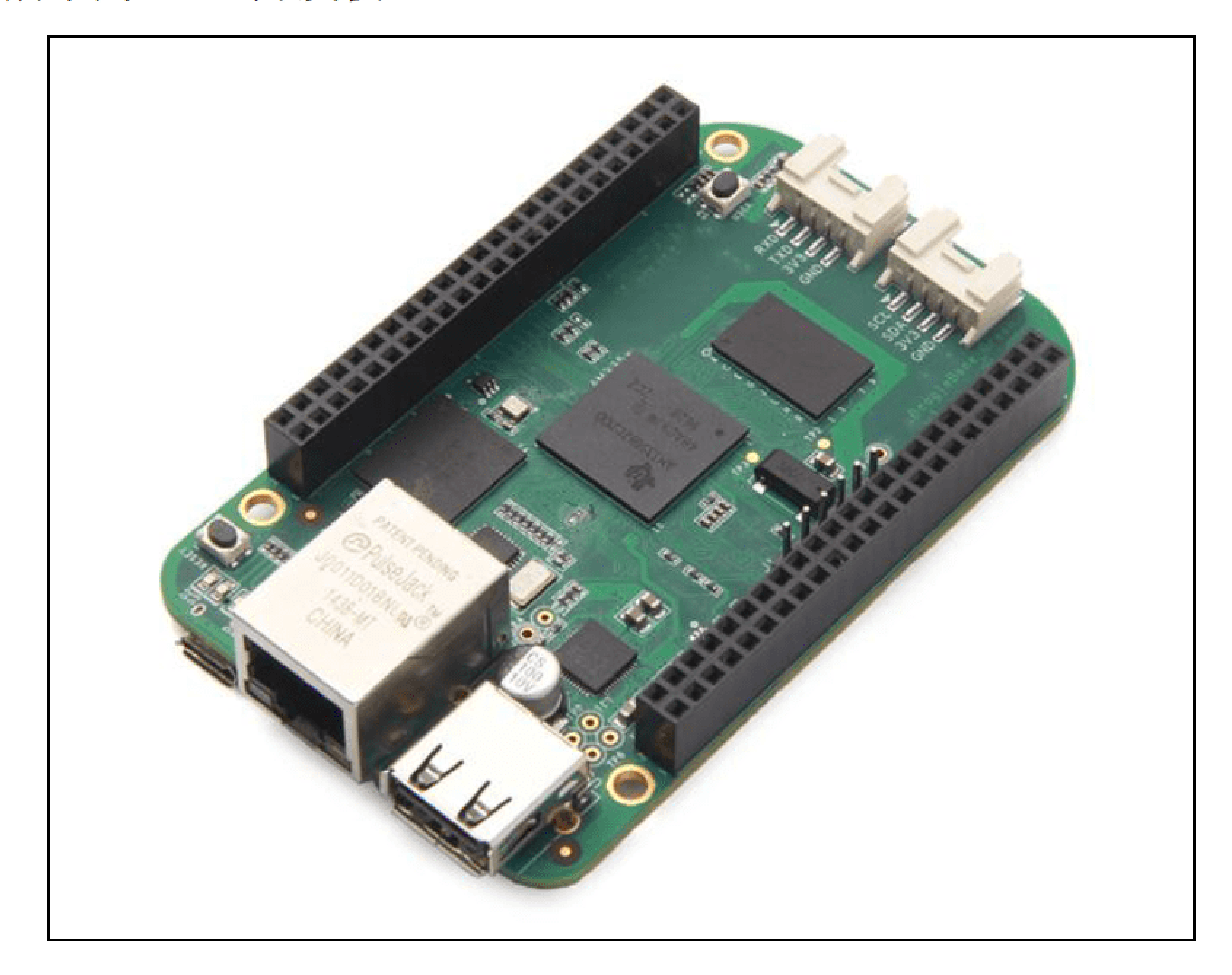
来源：http://www.exp-tech.de/beaglebone-black-rev-c-element14

图 1-11

SeeedStudio BeagleBone Green（BBG）是 BeagleBoard.org 和 Seeed Studio 合作研发的产品。BBG 有和 BBB 相同的特性，只是 HDMI 接口被 Grove 连接器替代，如此一来 BBG 的价格比 BBB 更低。可以通过网址 http://www.seeedstudio.com/depot/SeeedStudio-

BeagleBone-Green-p-2504.html 浏览并购买该开发板。

图 1-12 所示为 BBG 开发板。



来源：http://www.seeedstudio.com/depot/SeeedStudio-BeagleBone-Green-p-2504.html

图 1-12

基于 ESP8266 MCU 的物联网开发板

ESP8266 是一块集成 TCP/IP 的低成本的 Wi-Fi MCU。它由 Espressif Systems——一家中国制造商制造。可以通过网址 http://espressif.com/en/products/hardware/esp8266ex/overview 获取更多关于该芯片的信息。

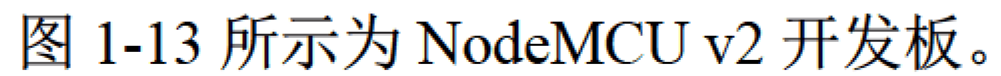
有很多基于 ESP8266 芯片的开发板。下面列出了基于 ESP8266 MCU 构建的开发板平台。

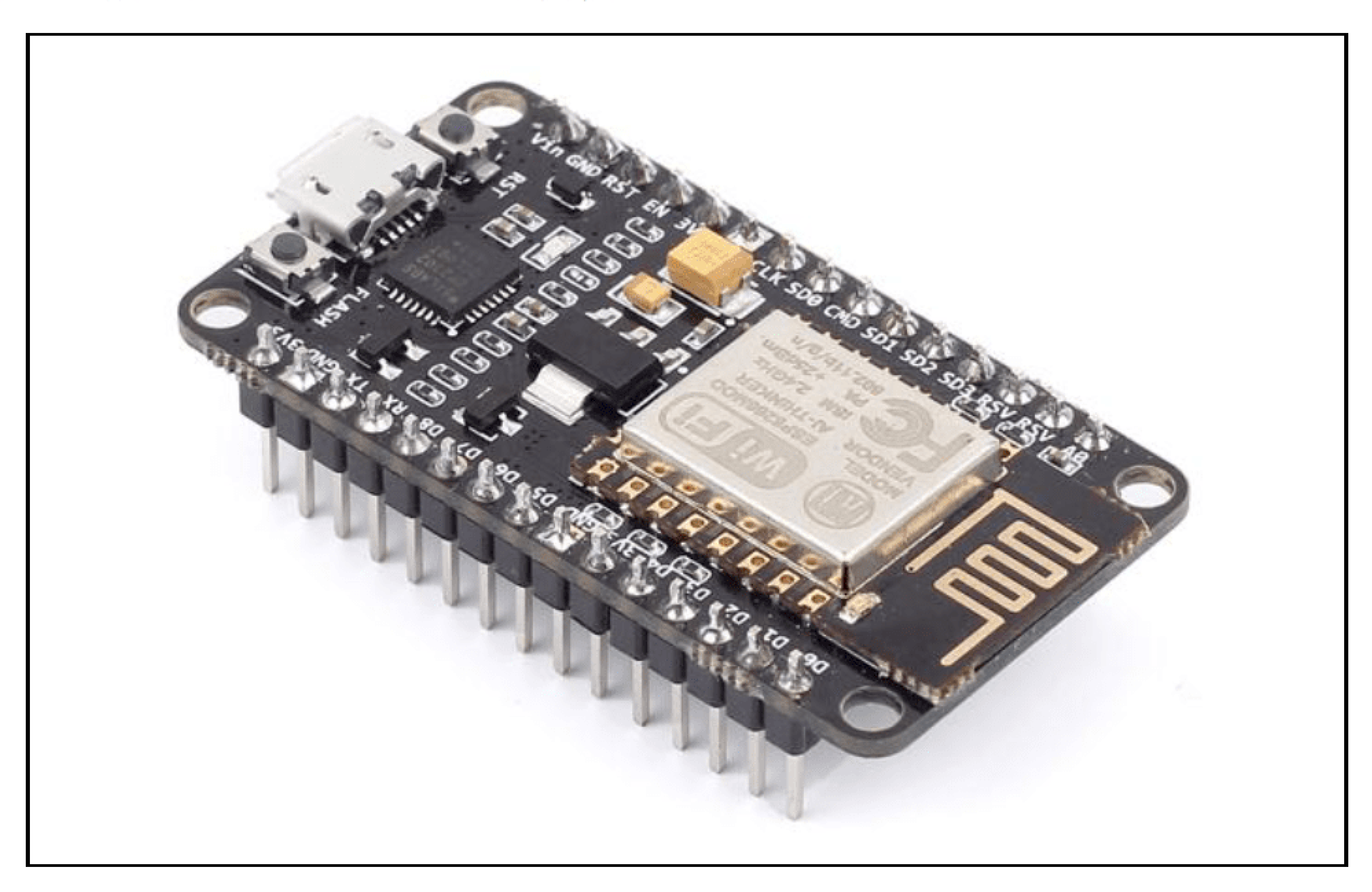
- NodeMCU：该开发板使用以 Lua 作为编程语言的 NodeMCU 固件。官方网址为 http://www.nodemcu.com/。
- SparkFun ESP8266 Thing：由 SparkFun 开发，应该使用串行硬件，如 FIDI 来将程序写入该开发板，但该产品兼容于 LiPo 充电器。可以通过网址 https://www.sparkfun.com/products/13231 阅读更多关于它的信息。
- SparkFun ESP8266 Thing-Dev：该开发板已经包含了 FTDI-to-USB 工具，但是没

有 LiPo 充电器。它由 SparkFun 开发，产品信息可以通过网址 https://www.sparkfun.com/products/13711 进行阅读。

- SparkFun Blynk board – ESP8266：该开发板包含温度和湿度感应装置。可以通过网址 https://www.sparkfun.com/products/13794 阅读更多信息。
- Adafruit HUZZAH with ESP8266 Wi-Fi：由 Adafruit 生产。产品信息可以通过网址 https://www.adafruit.com/products/2821 进行阅读。

如果对 ESP8266 芯片感兴趣，建议通过网址 http://www.esp8266.com 加入 ESP8266 论坛。


图 1-13 所示为 NodeMCU v2 开发板。

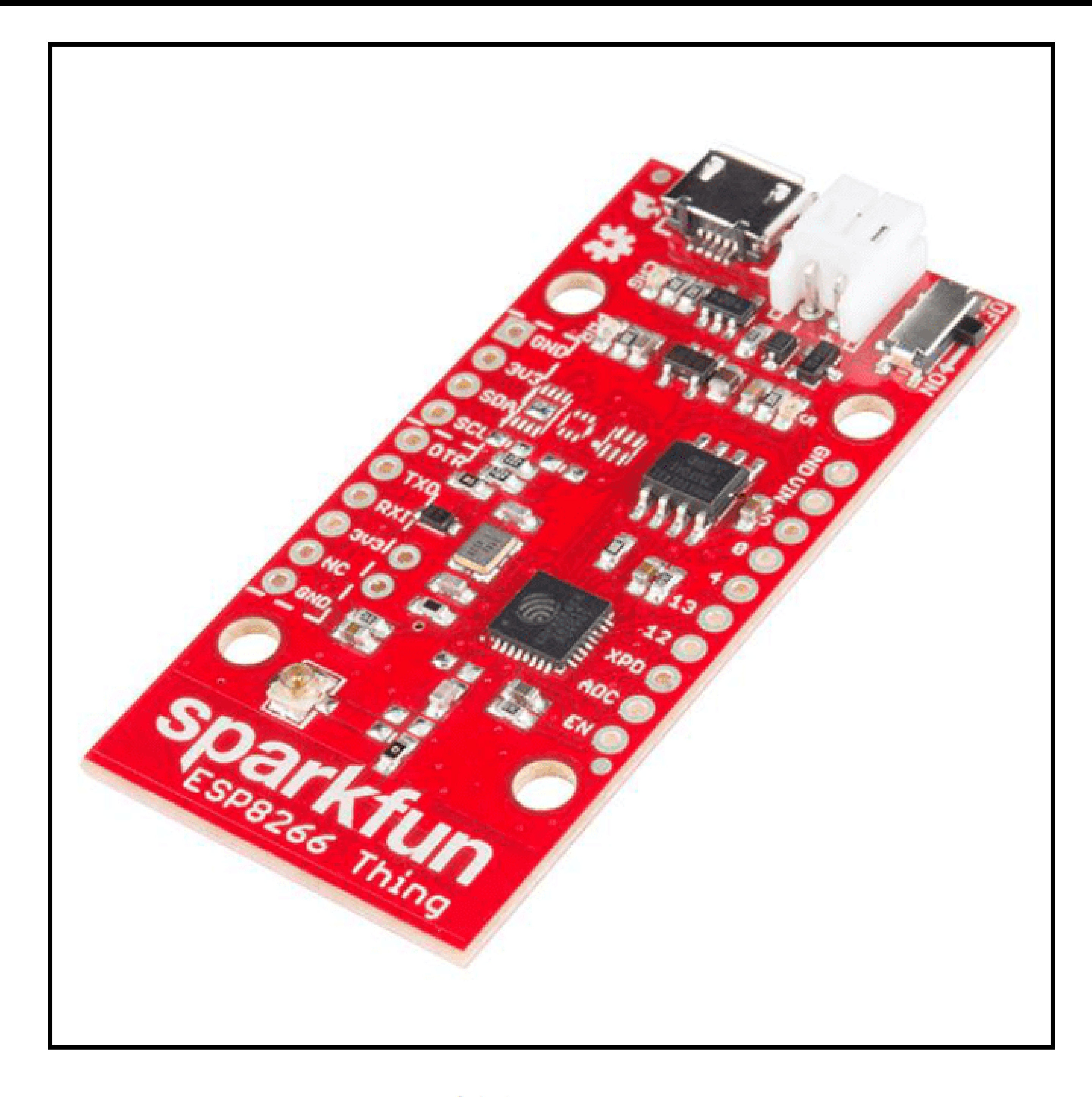


来源：http://www.seeedstudio.com/depot/NodeMCU-v2-Lua-based-ESP8266-development-kit-p-2415.html

图 1-13

尽管 NodeMCU v2 和 SparkFun ESP8266 Thing 开发板有相同的芯片，但它们的芯片模块不同。NodeMCU v2 使用 ESP8266 模块，SparkFun ESP8266 Thing 开发板使用 ESP8266EX 芯片。另外，SparkFun ESP8266 Thing 开发板提供一个可以附到外接电池的 LiPo 连接口。

图 1-14 所示为一块 SparkFun ESP8266 Thing 开发板。



SparkFun ESP8266 Thing board. 来源：https://www.sparkfun.com/products/13231

图 1-14

基于 TI CC3200 MCU 的物联网开发板

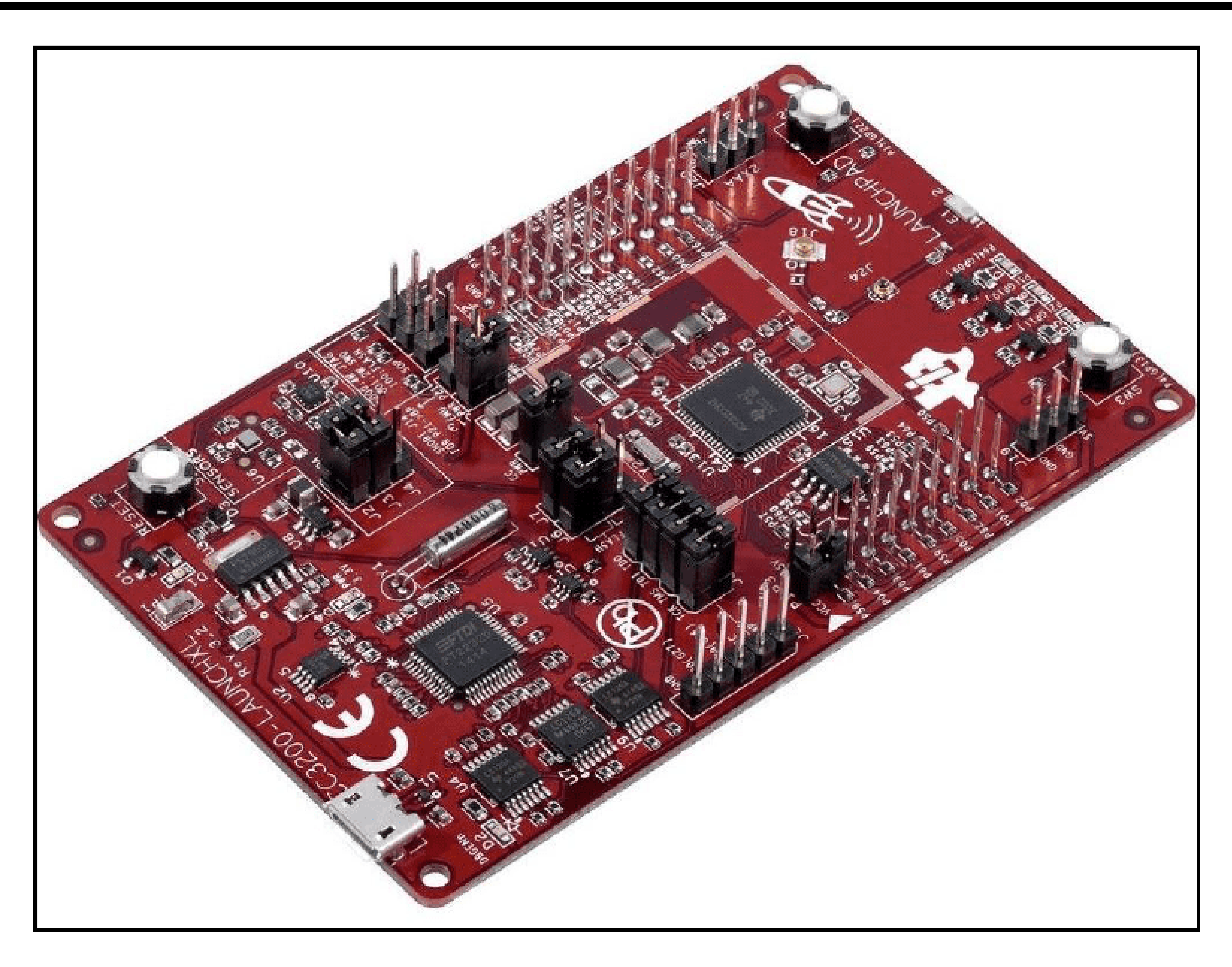
TI CC3200 是一块来自德州仪器的基于 ARM® Cortex®-M4 的 Wi-Fi MCU。该开发板是一个对于物联网的完全解决方案。该芯片支持站点、接入点和 Wi-Fi 直接模块。在安全性方面，TI CC3200 支持 WPA2 个人和企业安全还有 WPS 2.0。可以通过网址 http://www.ti.com/product/cc3200 浏览该模块。

对于物联网研发，德州仪器提供了 SimpleLink Wi-Fi CC3200 LaunchPad 评测工具，这是一个用于研发和调试的完善的工具包。

图 1-15 所示为一块 SimpleLink Wi-Fi CC3200 LaunchPad 开发板。

TI CC3200 也被 Readbear 所使用（http://redbear.cc），用于研发 RedBearLab CC3200 和 RedBearLab Wi-Fi Micro 开发板。这些开发板具有和 SimpleLink Wi-Fi CC3200 LaunchPad 开发板相同的功能，但是排除了 CC3200 调试工具。这些开发板的价格也同样比 SimpleLink Wi-Fi CC3200 LaunchPad 开发板低。

图 1-16 所示为一块 RedBearLab CC3200 开发板。



来源：https://www.conrad.de/de/entwicklungsboard-texas-instruments-cc3200-launchxl-1273804.html

图 1-15



来源：http://www.exp-tech.de/redbearlab-cc3200

图 1-16

物联网设备感知和启动

本节将学习物联网设备如何感知和启动。这部分非常重要，因为可以通过感知设备或者用启动设备和环境交互来收集数据。测试时，会使用 Arduino（Arduino.cc）和 Raspberry Pi 开发板。

Arduino 设备感知和启动

大部分 Arduino 研发使用 Sketch，如图 1-17 所示。这是一个编写 Arduino 应用的简单编程语言。具有 C/C++经验的读者，会对它的编程语法感到很熟悉。

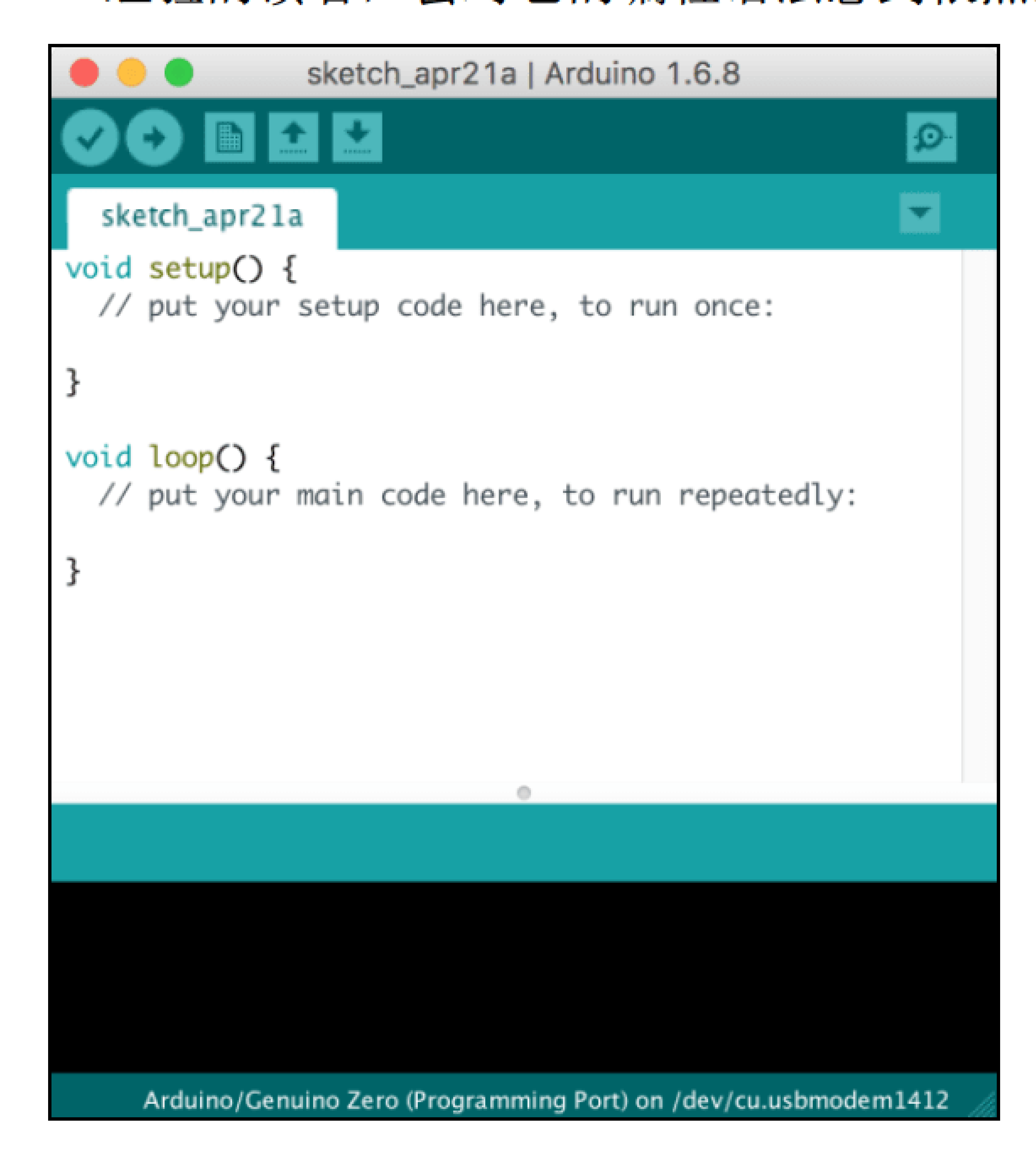


图 1-17

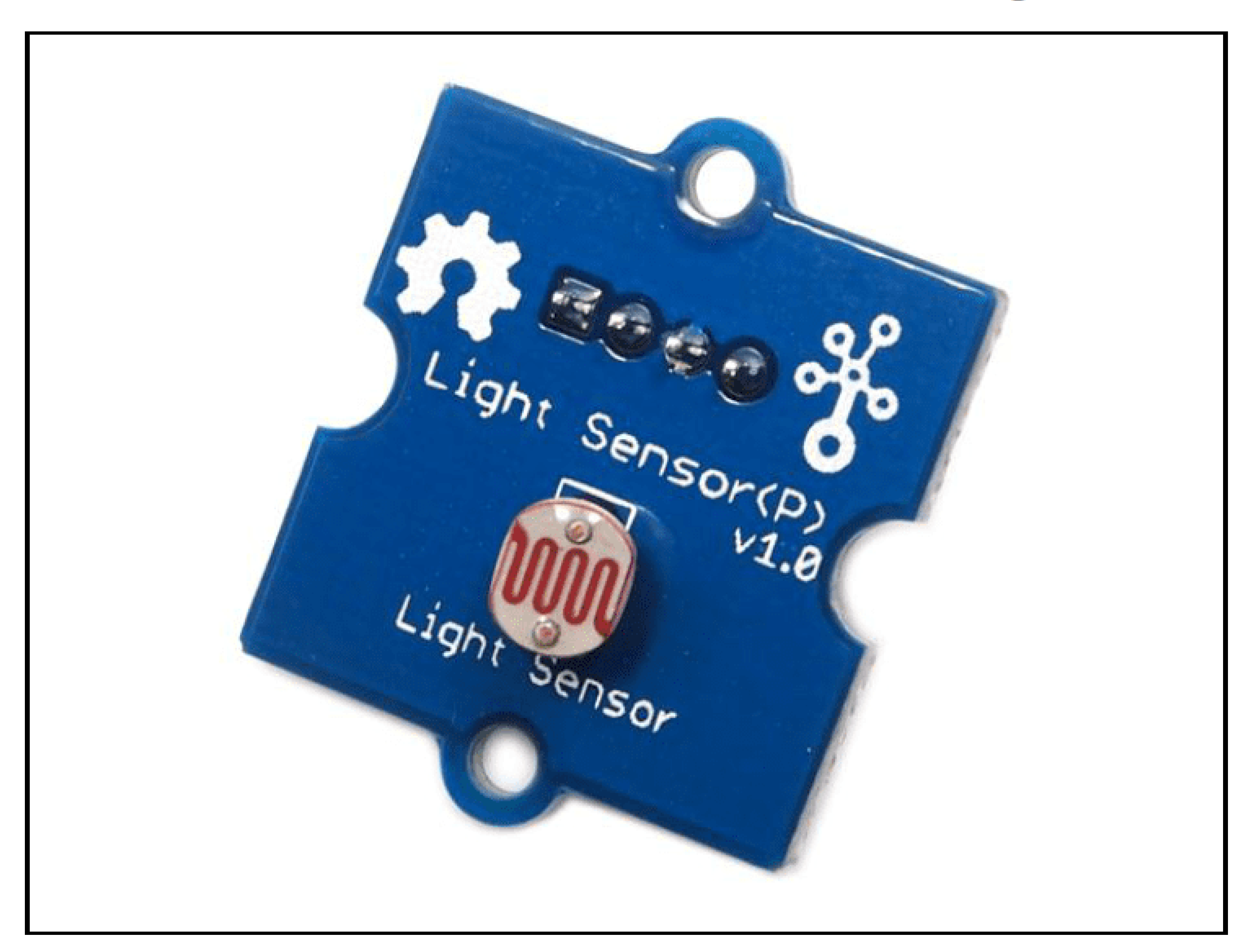
要开始研发，可以从网址 https://www.arduino.cc/en/Main/Software 下载软件。之后，可以通过网址 http://www.arduino.cc/en/Reference/HomePage 阅读更多关于 Arduino API 的信息。

跟电子和模拟 I/O 相关，应该熟悉下列 Sketch API：

- digitalRead()从 Arduino 上的电子针内读数据。

- digitalWrite()向 Arduino 上的电子针内写入数据。
- analogRead()从 Arduino 上的模拟针内读模拟数据。
- analogWrite()向 Arduino 上的电子针内写入模拟数据。

本节将会展示如何在 Arduino 上跟模拟和电子 I/O 工作，会用到一个光感应器。LDR 或者 Photoresistor 感应器是很廉价的光感应设备，甚至 Seeedstudio 已经设计了一个模块形式的光感应设备 Grove – Light Sensor(P)。可以通过网址 http://www.seeedstudio.com/depot/Grove-Light-SensorP-p-1253.html 阅读更多信息。该模块有 4 个脚针，但是只能连接 Arduino 开发板上的 VCC 和 GND 脚针到对应的 VCC 和 GND 脚针。然后，SIG 脚针连接到 Arduino 开发板上的模拟脚针。图 1-18 展示了 Grove – Light Sensor(P)。



来源：http://www.seeedstudio.com/depot/Grove-Light-SensorP-p-1253.html

图 1-18

将会需要下列资源用于测试：

- 跨接电缆。
- 光感应模块，http://www.seeedstudio.com/depot/Grove-Light-SensorP-p-1253.html。

现在可以将键 6 和键 7 相连。然后，LDR 模块的 SIG 脚针连接到 A0。LDR 模块的 VCC 和 GND 脚针被连接到 3V3 和 GND。可以通过图 1-19 看到硬件布线情况。

布线是由 Fritzing 实现的（http://fritzing.org），适用于 Windows、Linux 和 Mac。可以用一些电子部件和开发板来实现自己的布线。该工具提供了很多电子部件，也可以添加自己的电子部件或者从互联网上下载。此处的布线实现如图 1-20 所示。

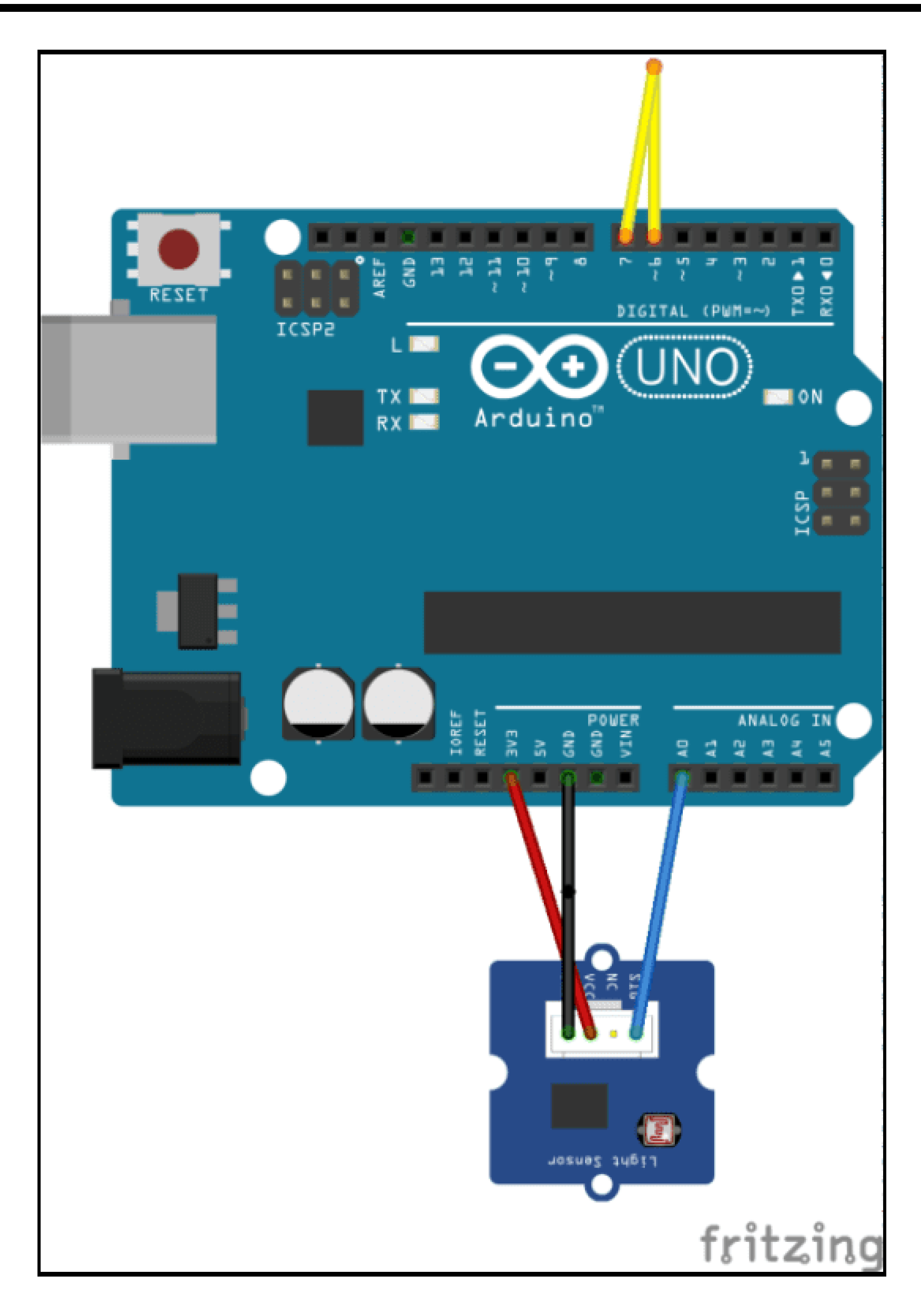

RESET
ICSP2
AREF
GND
DIGITAL (PWM=~)
L
TX
RX
UNO
Arduino
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
fritzing

图 1-19

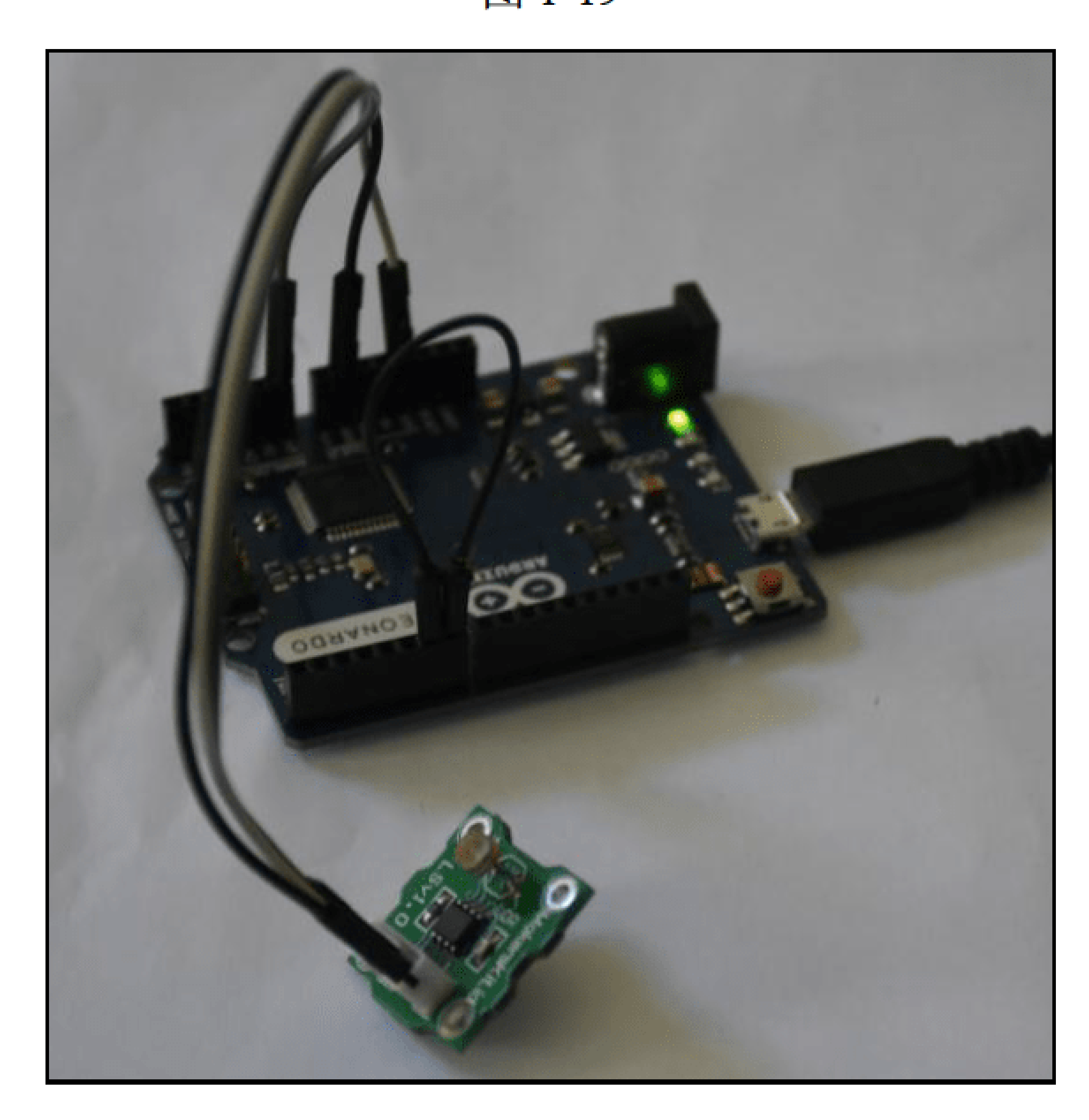


图 1-20

下一步是编写一个 Arduino 程序。打开 Arduino 软件，然后写入下列程序：

```
int dig_output = 7;
int dig_input = 6;
int analog input = A0;

int digital val = LOW;

void setup() {
  Serial.begin(9600);

  pinMode(dig_output, OUTPUT);
  pinMode(dig_input, INPUT);
}
void loop() {

  digitalWrite(dig_output,digital_val);
  int read_digital = digitalRead(dig_input);
  Serial.print("Digital write: ");
  Serial.print(digital val);
  Serial.print(" read: ");
  Serial.println(read_digital);

  int ldr = analogRead(analog_input);
  Serial.print("Analog read: ");
  Serial.println(ldr);

  if(digital val==LOW)
    digital_val = HIGH;
  else
    digital val = LOW;

  delay(1000);
}
```

将该程序存储为 ArduinoIO。可以通过 Arduino 软件将该程序部署到 Arduino 开发板上。

当这些都完成时，可以从 Arduino 软件打开 Serial Monitor 工具，如图 1-21 所示。

该程序从电子脚针 7 到电子脚针 6 发送电子数据（高或者低值）。可以通过 analogRead()在 A0 脚针上从 LDR 模块读入亮度值。

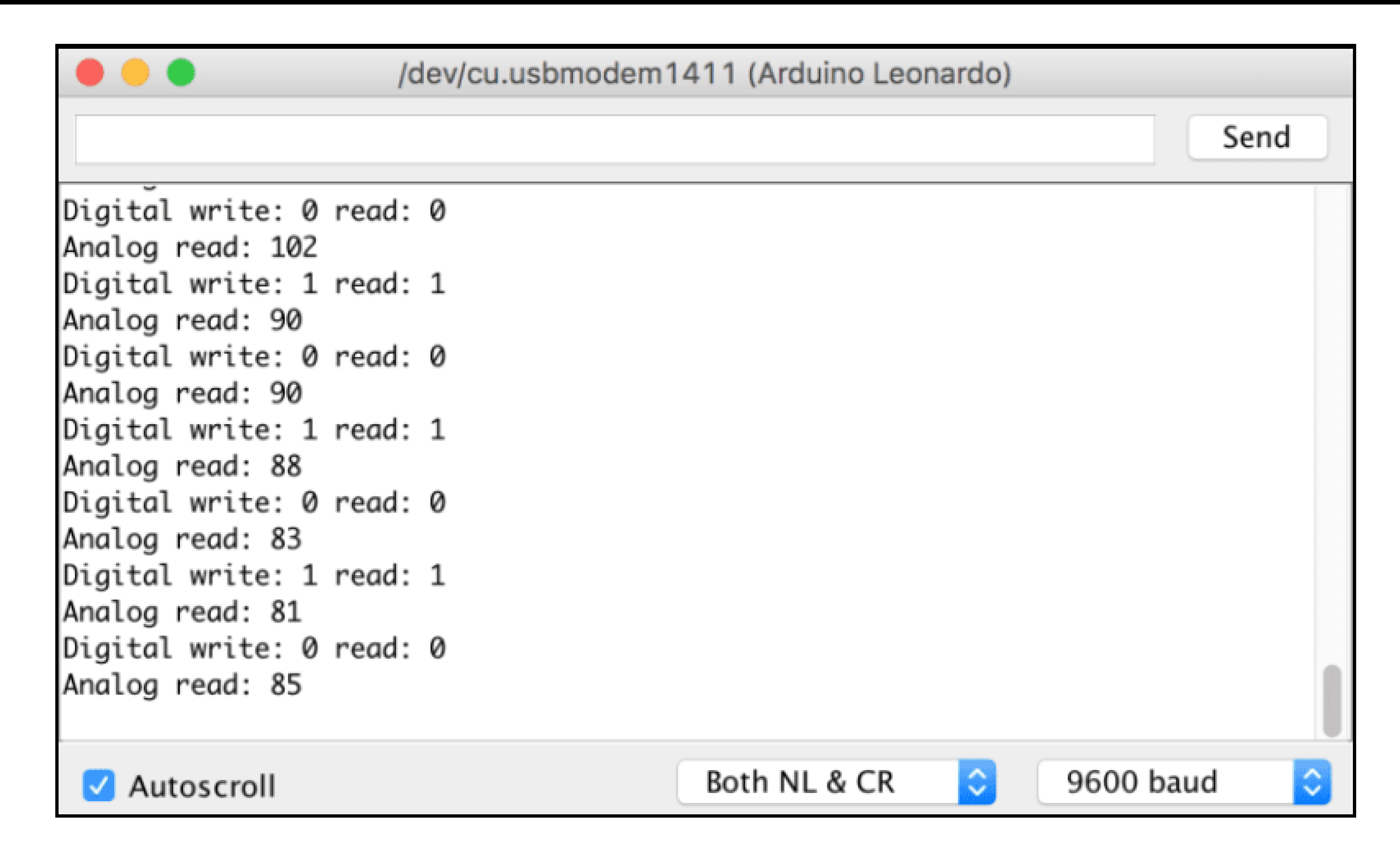
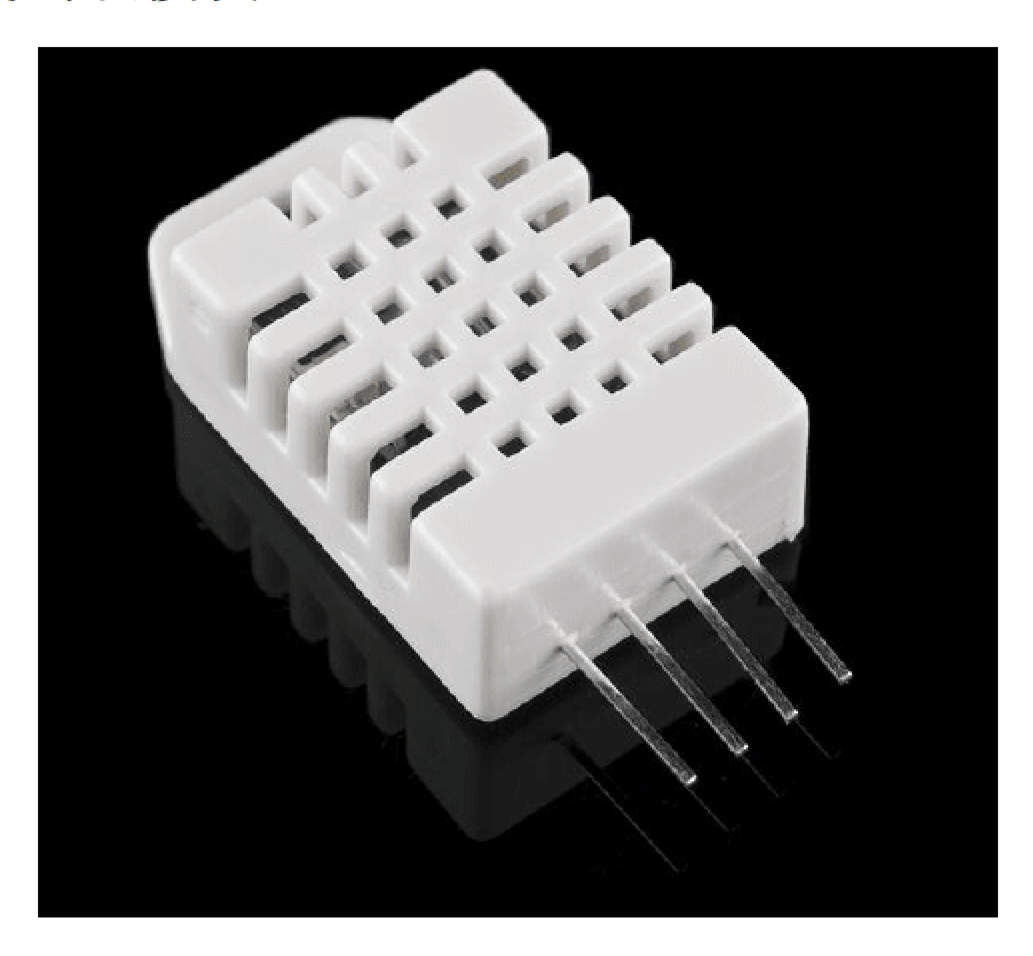


图 1-21

对于第二个感应器测试情境，将尝试从一个感应设备读取温度和湿度数据。这里使用 DHT22，如图 1-22 所示。RHT03（也叫作 DHT22）是一个低成本带有单线电子接口的湿度和温度感应器。可以从 SparkFun 获得该模块，网址为 https://www.sparkfun.com/products/10167，还有 Adafruit，网址为 https://www.adafruit.com/products/393。也可以在本地或者在线电子商店找到该模块。



DHT22 模块。来源：https://www.adafruit.com/products/385

图 1-22

如果想了解更多关于 DHT22 模块的信息，可以通过网址 http://cdn.sparkfun.com/datasheets/Sensors/Weather/RHT03.pdf 阅读 DHT22 数据库。

现在连接 DHT22 模块至 Arduino。下面列出了所有接线：

- ❑ VDD（脚针 1）在 Arduino 上被连接到 V3V 脚针。
- ❑ SIG（脚针 2）在 Arduino 上被连接到电子脚针 8。
- ❑ GND（脚针 4）在 Arduino 上被连接到 GND。

可以在图 1-23 中看到该布线。

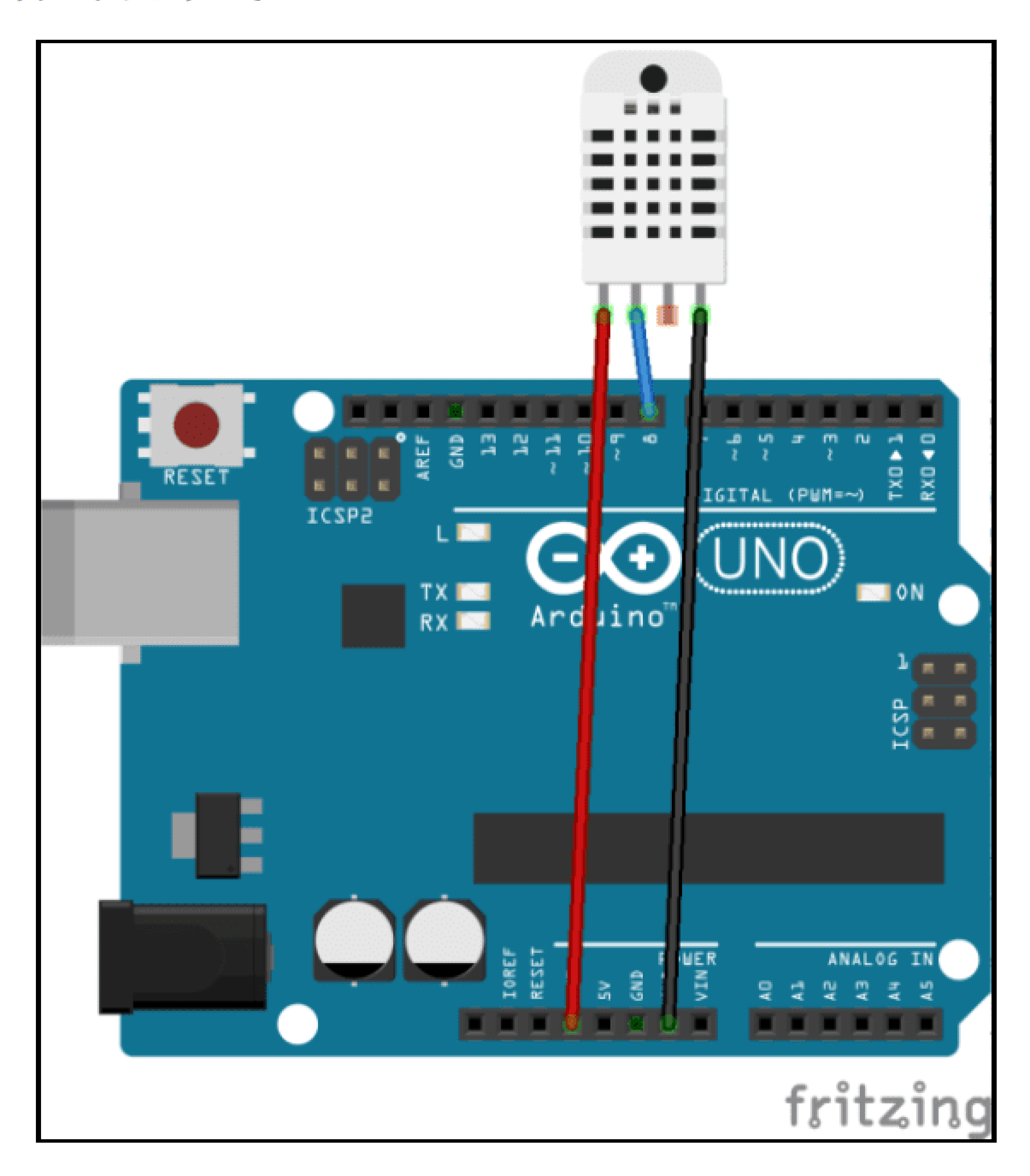


图 1-23

可以在图 1-24 中看到布线实现。

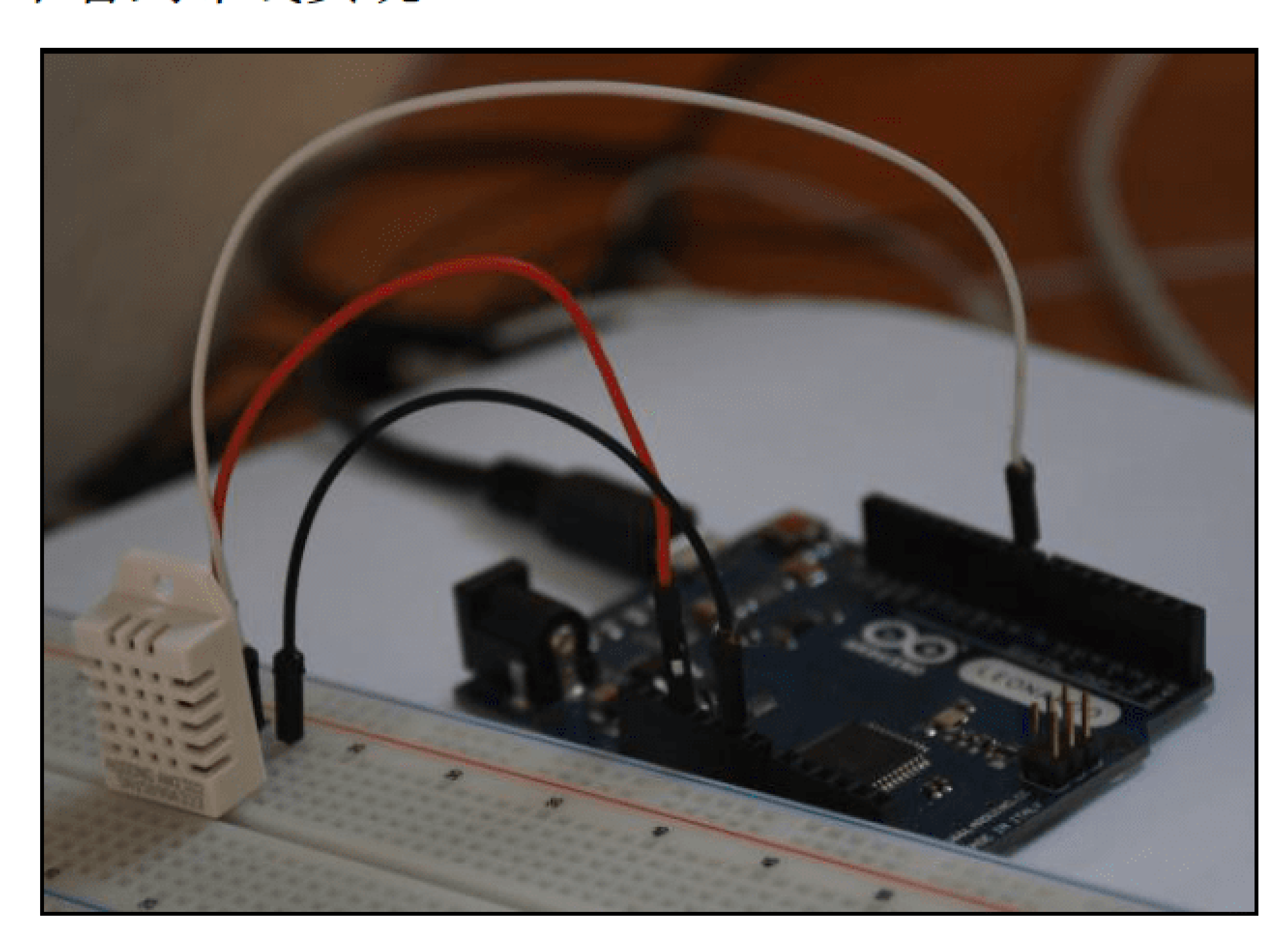


图 1-24

为了在 Arduino 上使用 DHT22，可以用由 Adafruit 提供的 DHT Sensor 函数库 https://github.com/adafruit/DHT-sensor-library。可以从 Arduino 软件里安装该函数库。选择菜单 Sketch | Include Library | Manage Libraries，会看到下列对话框。

在 Library Manager 里搜索 dht，可以看到 Adafruit 的 DHT Sensor 函数库，如图 1-25 所示。安装该函数库。

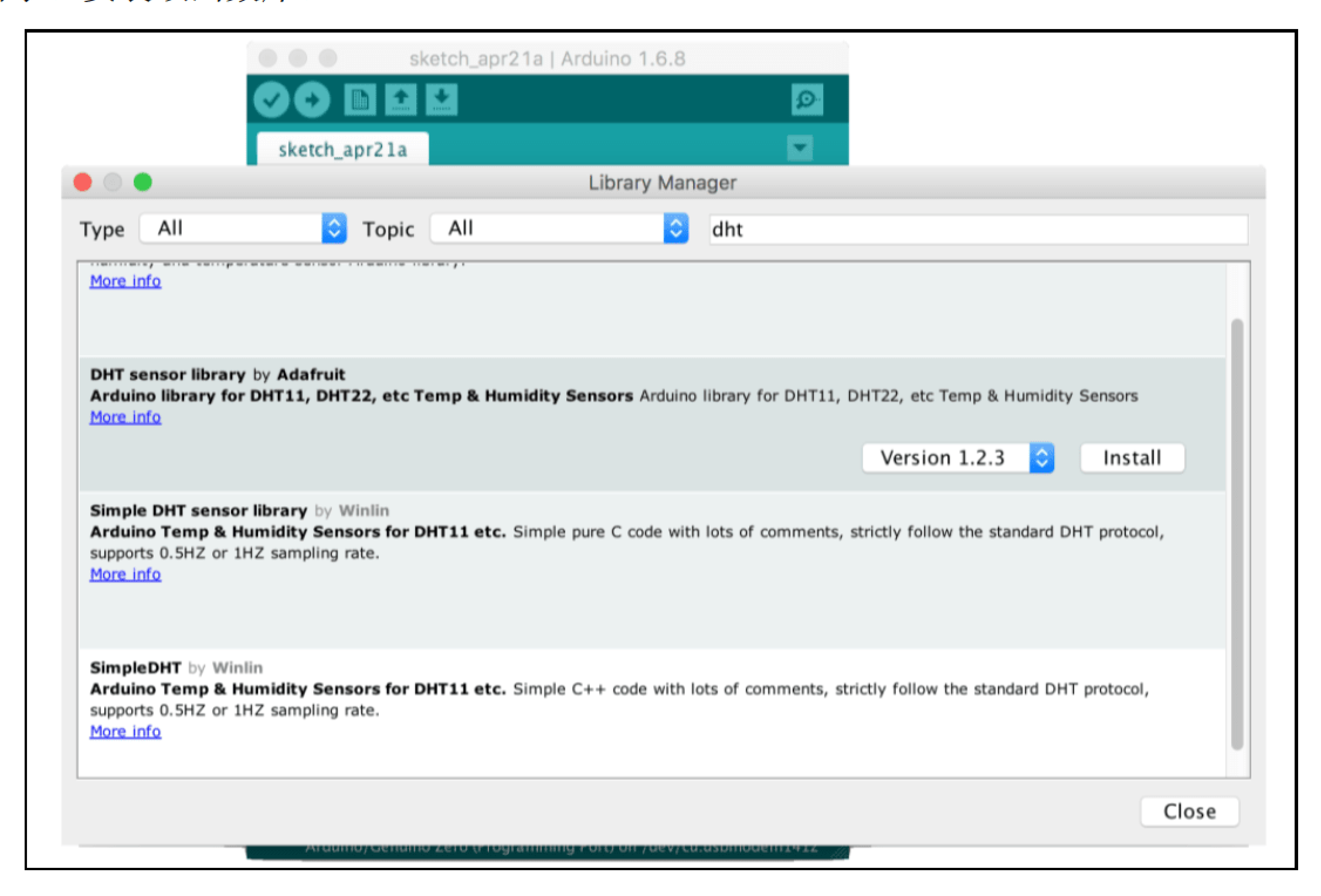


图 1-25

当安装完成后，可以开始在 Arduino 软件上写程序。下面是 DHT22 模块上的用于读取温度和湿度的程序样本：

```
#include "DHT.h"

//定义 DHT22
#define DHTTYPE DHT22
//定义 DHT22 上的 pin
#define DHTPIN 8
DHT dht(DHTPIN, DHTTYPE);

void setup() {
```

```
  Serial.begin(9600);
  dht.begin();
}

void loop() {
  delay(2000);

  //读入温度或湿度用时大概 250ms
  //检测器读入可能耗时最多 2s（这是一个非常老的检测器）
  float h = dht.readHumidity();
  //以摄氏度标准读取温度（默认）
  float t = dht.readTemperature();


  //检查是否有任何读取失败和提前退出（再试一次）
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  //以摄氏度标准读取热指数（isFahreheit = false）
  float hic = dht.computeHeatIndex(t, h, false);

  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" *C\t");
  Serial.print("Heat index: ");
  Serial.print(hic);
  Serial.println(" *C ");
}
```

将该程序保存为 ArduinoDHT。现在可以将该程序编译并且上传到 Arduino 开发板。之后，打开 Serial Monitor 来观察温度和湿度数据，如图 1-26 所示。

它是如何工作的？

在 setup()函数中，通过调用 dht.begin()初始化 DHT 模块。为了读入温度和湿度，可以使用 dht.readTemperature()和 dht.readHumidity()，也可以使用 dht.computeHeatIndex()函数得到热指数。

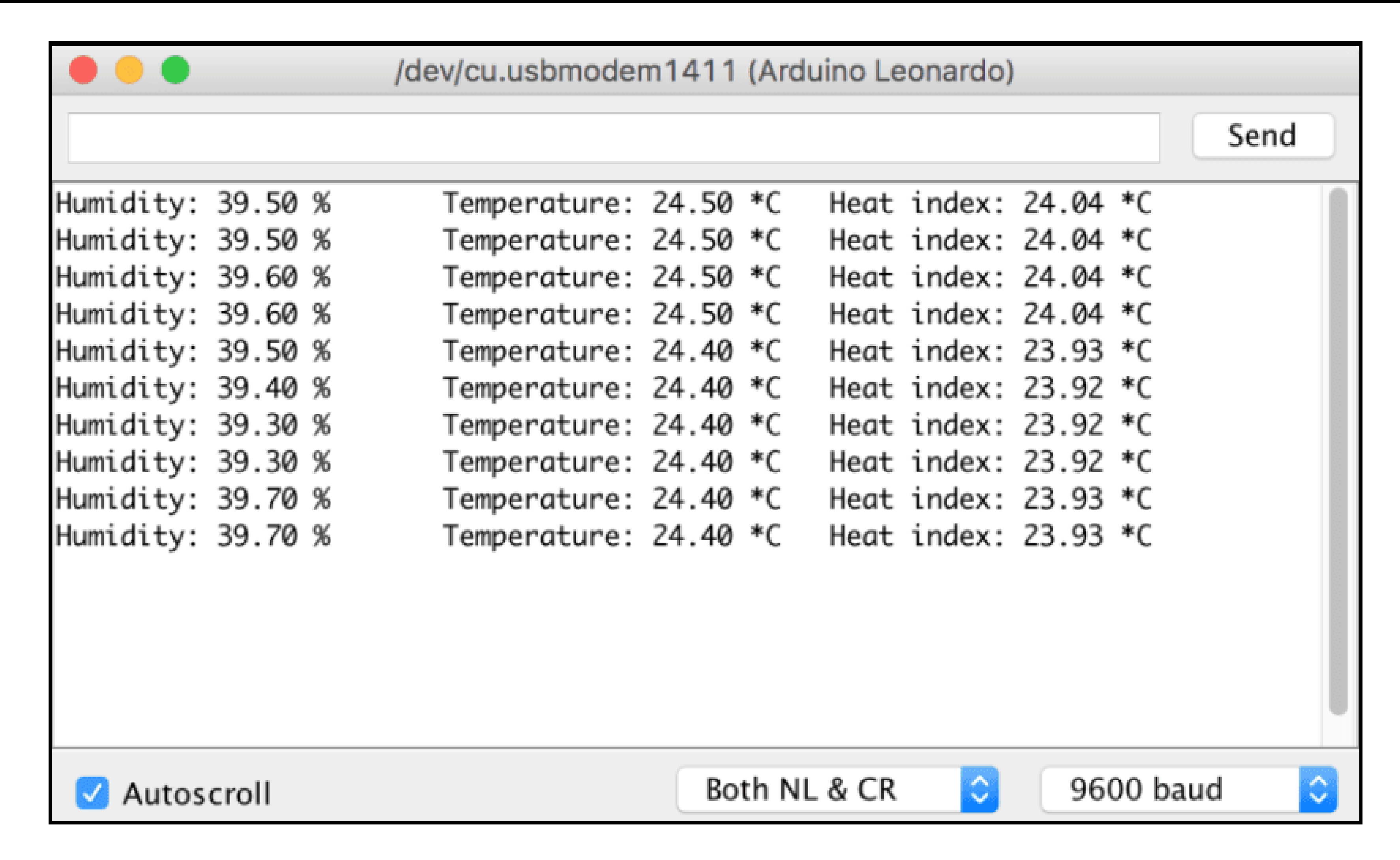


图 1-26

Raspberry Pi 设备感知和启动

Raspberry Pi 开发板是本书中用于测试实验的开发板之一。在本节中，使用 Raspberry Pi 来和外部设备一起感知和启动。这里会使用一块 Raspberry Pi 3 开发板来做测试。

配置

在使用 Raspberry Pi 开发板之前，需要在开发板上配置一个操作系统（OS）。OS 软件可以在 microSD 卡上配置。推荐使用一块 8GB 的 microSD 卡。有很多可以在 Raspberry Pi 开发板上使用的 OS 软件，可以通过网站 https://www.raspberrypi.org/downloads/查看更多信息。

为了测试，这里会使用 Raspbian（https://www.raspberrypi.org/downloads/raspbian/）作为 Raspberry Pi 开发板的操作系统。Raspbian 是基于 Debian 的操作系统，为 Raspberry Pi 优化。按照网址 https://www.raspberrypi.org/documentation/installation/installing-images/README.md 上的安装指南进行安装。Raspbian 仅仅是 Raspberry Pi OS 的其中一种操作系统。可以通过网址 https://www.raspberrypi.org/downloads/尝试其他 Raspberry Pi OS。

使用 Raspberry Pi GPIO

如果正在使用最新版本的 Raspbian（Jessie 或之后的），wiringPi 模块（http://wiringpi.com）已经为你安装好。可以通过如下命令在 Raspberry Pi Terminal 上验证 wiringPi 版本：

```
$ gpio -v
```

可以看到 wiringPi 模块版本。图 1-27 展示了该程序输出的一个样例。

```
Documents — pi@raspberrypi: ~ — ssh pi@192.168.0.12 — 80×17
pi@raspberrypi:~ $ gpio -v
gpio version: 2.32
Copyright (c) 2012-2015 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty

Raspberry Pi Details:
  Type: Pi 3, Revision: 02, Memory: 1024MB, Maker: Sony
  * Device tree is enabled.
  * This Raspberry Pi supports user-level GPIO access.
    -> See the man-page for more details
    -> ie. export WIRINGPI_GPIOMEM=1
pi@raspberrypi:~ $
```

图 1-27

另外，可以使用如下命令验证 Raspberry GPIO 布局：

```
$ gpio - readall
```

该命令将会显示 Raspberry Pi 的布局，可以检测 Raspberry Pi 模块。该程序从开发板 Raspberry Pi 3 上的输出样例可在图 1-28 中看到。

```
Documents — pi@raspberrypi: ~ — ssh pi@192.168.0.12 — 80×28
pi@raspberrypi:~ $ gpio readall
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 |   IN | 1 |  3 || 4  |   |      | 5V      |     |     |
 |   3 |   9 |   SCL.1 |   IN | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 1 | ALT5 | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | ALT5 | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI |   IN | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO |   IN | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK |   IN | 0 | 23 || 24 | 1 | IN   | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | IN   | CE1     | 11  | 7   |
 |   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
 |   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
 |   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
 |  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
 |  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
 |  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
 |     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
pi@raspberrypi:~ $
```

图 1-28

对于 Raspberry Pi GPIO 开发，最新的 Raspbian 也已经为 Python 安装了 RPi.GPIO 函数库，可从 https://pypi.python.org/pypi/RPi.GPIO 查看，所以可以直接使用它。

为了测试 Raspberry Pi GPIO，需要把 LED 安装到 GPIO11（BCM 17）上。可以在图 1-29 中看到布线结构。

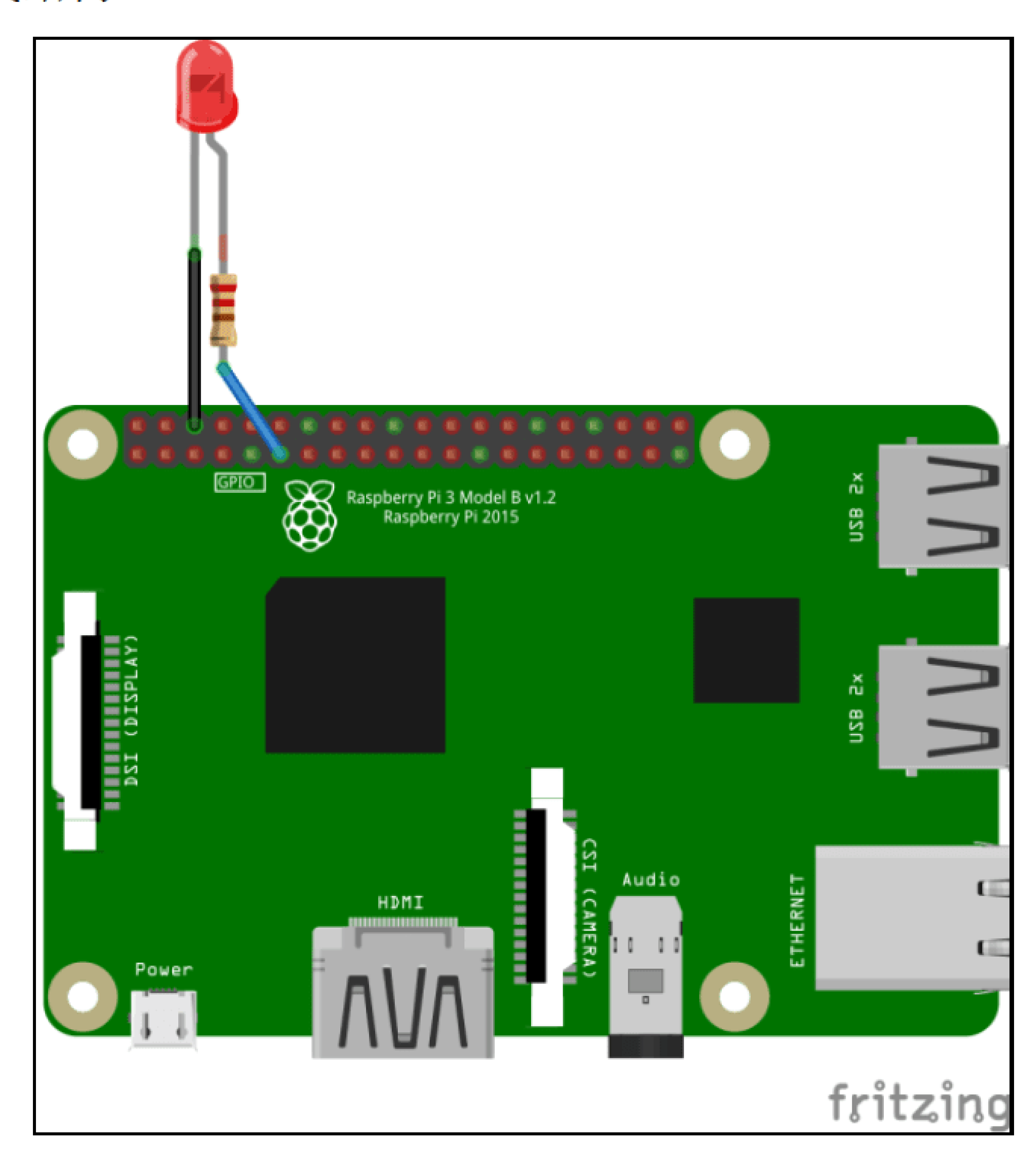


图 1-29

现在可以用自己的编辑器编写一个 Python 程序。请写下如下程序：

```
import RPi.GPIO as GPIO
import time


led_pin = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(led_pin, GPIO.OUT)


try:
    while 1:
        print("turn on led")
        GPIO.output(led_pin, GPIO.HIGH)
        time.sleep(2)
        print("turn off led")
        GPIO.output(led_pin, GPIO.LOW)
```

```
        time.sleep(2)

except KeyboardInterrupt:
    GPIO.output(led_pin, GPIO.LOW)
    GPIO.cleanup()


print("done")
```

下面是该代码的一些解释：

- 通过 GPIO.setmode（GPIO.BCM）来设置 GPIO 类型。将使用 GPIO.BCM 模式。在 GPIO BCM 中，应该可以从 GPIO 布局中看到 BCM 列上的 GPIO 数值。
- 定义 GPIO 作为输出模式，GPIO 将通过调用 GPIO.setup()来被使用。
- 为了设置电子输出，可以调用 GPIO.output()。GPIO.HIGH 是用来发送 1 到电子输出。否则，GPIO.LOW 将被用来发送 0 到电子输出。

将该程序保存并命名为 ch01_led.py。

现在可以通过在 Raspberry Pi Terminal 上输入下列命令来运行该程序：

```
$ sudo python ch01_led.py
```

由于权限问题，用 sudo 来执行该程序。为了访问 Raspberry Pi 硬件 I/O，需要本地管理员权限。

将会看到一个发光的 LED 并且从该程序处得到一个返回值。该程序输出的一个返回值可在图 1-30 中看到。

```
Documents — pi@raspberrypi: ~/Documents/book — ssh pi@192.168.0.12 — 8...
pi@raspberrypi:~ $ cd Documents/book/
pi@raspberrypi:~/Documents/book $ python ch01_led.py
turn on led
turn off led
turn on led
turn off led
turn on led
turn off led
turn on led
turn off led
turn on led
turn off led
turn on led
```

图 1-30

通过传感设备来感知

本节将会探索如何从 Raspberry Pi 来感知。这里使用 DHT22 从它所处的环境来收集

温度和湿度输入。

为了能用 Python 访问 DHT22，使用 Adafruit Python DHT Sensor 函数库。可以通过网址 https://github.com/adafruit/Adafruit_Python_DHT 来浏览该模块。

需要一些必需的函数库以使用 Adafruit Python DHT Sensor 函数库。在 Raspberry Pi Terminal 中输入下列命令：

```
$ sudo apt-get update
$ sudo apt-get install build-essential python-dev
```

现在就可以下载并安装 Adafruit Python DHT Sensor 函数库了：

```
$ git clone https://github.com/adafruit/Adafruit_Python_DHT
$ cd Adafruit_Python_DHT/
$ sudo python setup.py install
```

完成后，可以开始搭建布线。将 DHT22 模块连接至下列接口：

- ❑ DHT22 pin 1（VDD）被连接到 Raspberry Pi 上的 3.3V pin。
- ❑ DHT22 pin 2（SIG）被连接到 Raspberry Pi 上的 GPIO23（见 BCM 列）pin。
- ❑ DHT22 pin 4（GND）被连接到 Raspberry Pi 上的 GND pin。

完整的布线如图 1-31 所示。

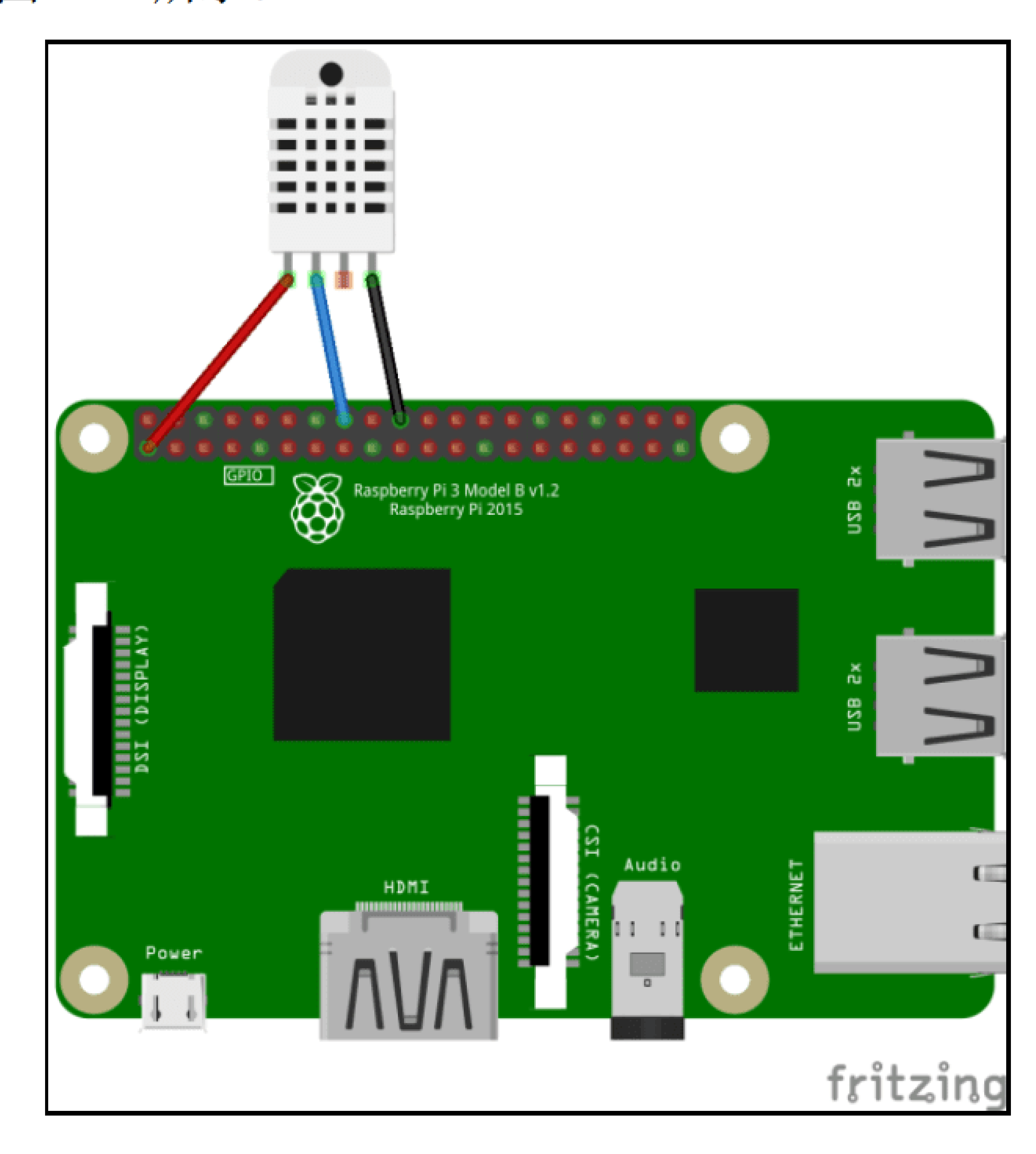


图 1-31

下一步就是写一个 Python 程序。可以写出如下代码：

```
import Adafruit_DHT
import time

sensor = Adafruit_DHT.DHT22

# DHT22 pin on Raspberry Pi
pin = 23


try:
    while 1:
        print("reading DHT22...")
        humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)

        if humidity is not None and temperature is not None:
            print('Temp={0:0.1f}*C
Humidity={1:0.1f}%'.format(temperature, humidity))

        time.sleep(2)

except KeyboardInterrupt:
    print("exit")


print("done")
```

将该程序保存为 ch01_dht22.py。然后，可以在 Raspberry Pi Terminal 上运行。请输入如下命令：

```
$ sudo python ch01_dht22.py
```

该程序的一个输出样本如图 1-32 所示。

它是如何工作的？

首先，通过调用 Adafruit_DHT.DHT22 对象来设置 DHT 模块类型，设置哪个 DHT22 pin 被附到 Raspberry Pi 开发板上。在本示例中，会使用 GPIO23 (BCM)。

为了获取温度和湿度检测数据，调用 Adafruit_DHT.read_retry(sensor, pin)。为了确保返回值不是 NULL，使用 conditional-if 来验证它们。

```
Documents — pi@raspberrypi: ~/Documents/book — ssh pi@192.168.0.12 — 8...
pi@raspberrypi:~/Documents/book $ sudo python ch01_dht22.py
reading DHT22...
Temp=30.2*C  Humidity=77.6%
reading DHT22...
Temp=30.2*C  Humidity=76.0%
reading DHT22...
Temp=30.1*C  Humidity=76.0%
reading DHT22...
Temp=30.1*C  Humidity=76.0%
reading DHT22...
Temp=30.2*C  Humidity=76.1%
reading DHT22...
^Cexit
done
pi@raspberrypi:~/Documents/book $
```

图 1-32

为房间建造一个智能温度控制器

为了控制房间的温度，可以建造一个智能温度控制器。在本例中，使用一个 PID（比例（proportional）-积分（integral）-导数（derivative））控制器。当设置一个特定的温度，PID 控制器将会通过制冷或制热来改变温度。PID 控制器程序使用 Python 开发，在 Raspberry Pi board 开发板上运行。

假设更冷和更热的机器通过继电器相连接。可以通过在继电器上发送 HIGH 信号来激活更冷或更热的机器。

让我们开始建造吧！

PID 控制器介绍

PID 控制是在工业街被广泛使用的最普遍的控制算法，并且已经被工业控制所普遍接受。PID 控制器之后的基本思想是读入检测器，然后通过计算比例（proportional）、积分（integral）和导数（derivative）返回并且对它们求和来计算期望的执行输出，并计算输出的 3 个部件。

一个典型 PID 控制器的设计样例如图 1-33 所示。

PID 控制器公式可按如下方式定义：

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}$$

$K_p K_i K_d$ 代表比例、积分和导数的系数。这些参数都是非负数值。变量 e 代表追踪误

差，追踪误差是所期望的输入值 i 和实际输出值 y 之间的差值。该误差信号 e 将被发送到 PID 控制器。

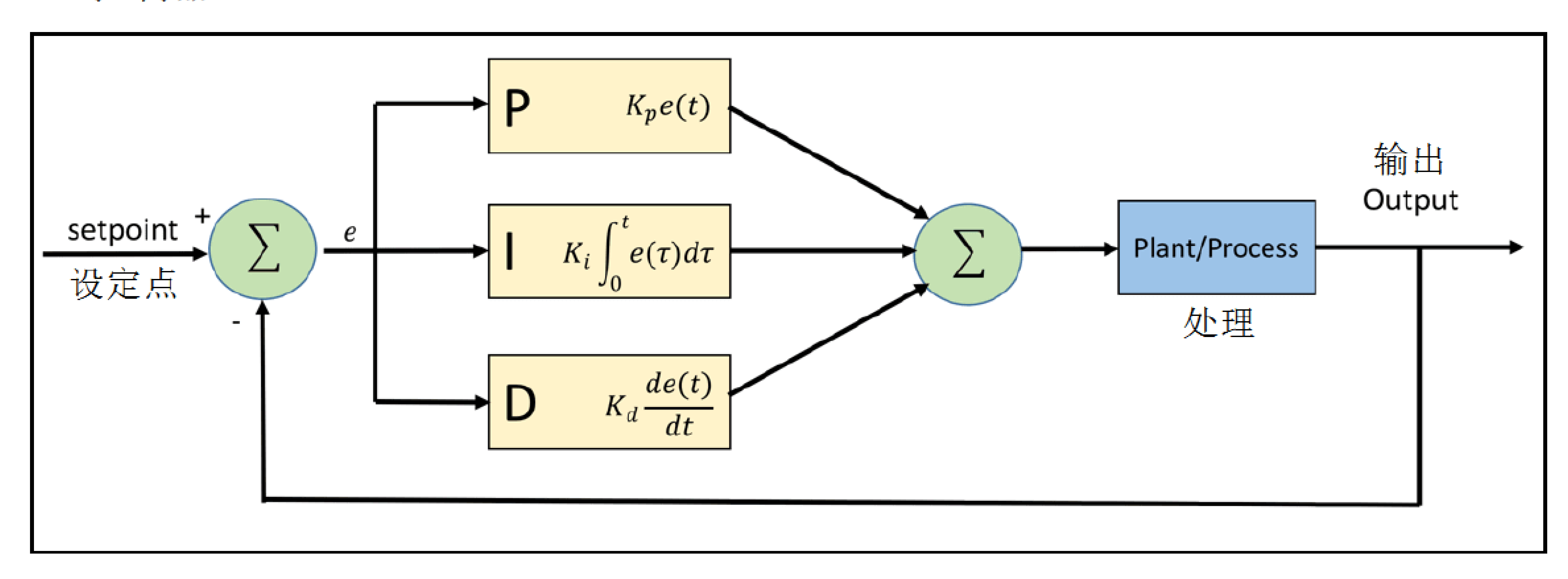


图 1-33

用 Python 实现 PID 控制器

本节将会搭建一个 Python 应用来实现 PID 控制器。大概来说，程序流程如图 1-34 所示。

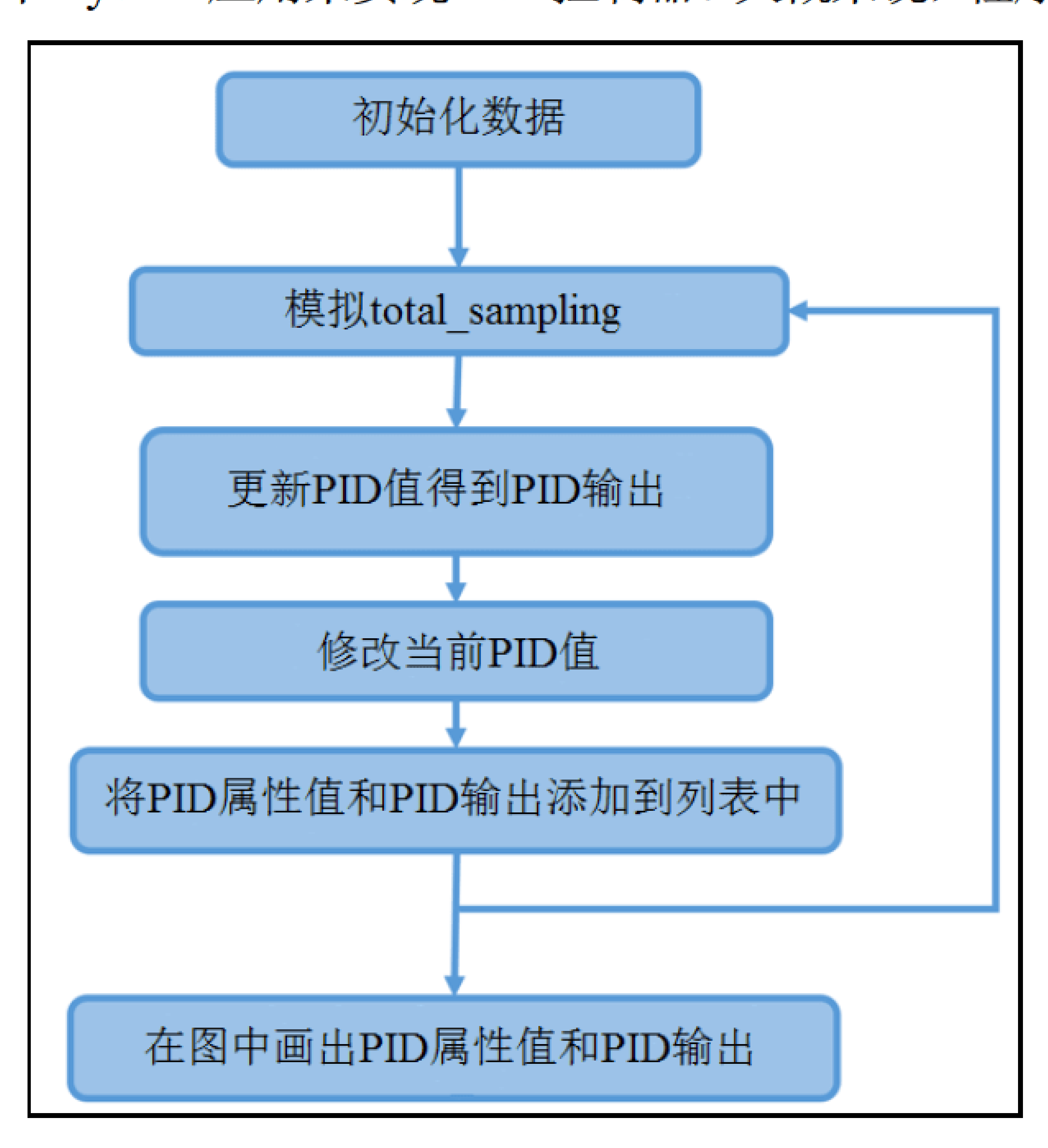


图 1-34

不应该完全从头开始搭建一个 PID 函数库。可以轻松地把 PID 控制器公式转化为 Python 代码。对于代码实现，会使用 https://github.com/ivmech/ivPID 中的 PID 类。下面是该文件的内容。

PID.py 文件：

```
import time

class PID:
    """PID Controller
    """

    def __init__(self, P=0.2, I=0.0, D=0.0):

        self.Kp = P
        self.Ki = I
        self.Kd = D

        self.sample_time = 0.00
        self.current_time = time.time()
        self.last_time = self.current_time

        self.clear()

    def clear(self):
        """清理 PID 计算和系数"""
        self.SetPoint = 0.0

        self.PTerm = 0.0
        self.ITerm = 0.0
        self.DTerm = 0.0
        self.last_error = 0.0

        # Windup Guard
        self.int_error = 0.0
        self.windup_guard = 20.0

        self.output = 0.0

    def update(self, feedback_value):
        """为给定的引用反馈计算 PID 数值
```

```
        .. 数学::
           u(t) = K p e(t) + K i \int {0}^{t} e(t)dt + K d {de}/{dt}

        .. 图片:: images/pid_1.png
           :align:   center

           用 Kp=1.2, Ki=1, Kd=0.001 (test_pid.py)测试 pid

        """
        error = self.SetPoint - feedback value

        self.current_time = time.time()
        delta time = self.current time - self.last time
        delta_error = error - self.last_error

        if (delta_time >= self.sample_time):
            self.PTerm = self.Kp * error
            self.ITerm += error * delta time

            if (self.ITerm < -self.windup_guard):
                self.ITerm = -self.windup_guard
            elif (self.ITerm > self.windup_guard):
                self.ITerm = self.windup guard

            self.DTerm = 0.0
            if delta_time > 0:
                self.DTerm = delta_error / delta_time

            # 记住上一次时间和误差用于下次计算
            self.last_time = self.current_time
            self.last error = error

            self.output = self.PTerm + (self.Ki * self.ITerm) + (self.Kd
* self.DTerm)

    def setKp(self, proportional gain):
        """通过设置 Proportional Gain 来决定 PID 应该多激进地对当前误差做出
反应 """
        self.Kp = proportional_gain
```

```
    def setKi(self, integral_gain):
        """通过设置 Integral Gain 来决定 PID 应该多激进地对当前误差做出反应"""
        self.Ki = integral_gain

    def setKd(self, derivative_gain):
        """通过设置 Derivative Gain 来决定 PID 应该多激进地对当前误差做出反应"""
        self.Kd = derivative_gain

    def setWindup(self, windup):
        """Integral windup，也被叫作 integrator windup 或者 reset windup，
        代表 PID 反馈中的一种情形，在该情形中，
        定位点上一个大的变化发生（假设是一个正向变化）
        并且积分项累积为一个大的误差
        在 rise(windup)过程中，因此超出而继续
        来增加，当该累积误差尚未受损
        （在另一个方向上的补偿项）
        该特定程序是过量超出。
        """
        self.windup_guard = windup

    def setSampleTime(self, sample_time):
        """应该以固定间隔被更新的 PID。
        基于预定义的样本时间，PID 决定它是否应该计算或者立即返回。
        """
        self.sample_time = sample_time
```

出于测试目的，创建一个简单程序用于模拟。需要依赖的函数库有 NumPy、SciPy、Pandas、Patsy，还有 matplotlib 函数库等。首先，应该安装 python-dev 用于 Python 开发。在 Raspberry Pi Terminal 中输入下列命令：

```
$ sudo apt-get update
$ sudo apt-get install python-dev
```

就绪之后，可以安装 NumPy、SciPy、Pandas 还有 Patsy 函数库。打开 Raspberry Pi Terminal 并且输入下列命令：

```
$ sudo apt-get install python-scipy
$ pip install numpy scipy pandas patsy
```

最后一步就是从源码安装 matplotlib 函数库。在 Raspberry Pi Terminal 中输入下列命令：

```
$ git clone https://github.com/matplotlib/matplotlib
$ cd matplotlib
$ python setup.py build
$ sudo python setup.py install
```

一旦这些所依赖的函数库安装后，可以测试 PID.py 文件。输入下列程序：

```
import matplotlib
matplotlib.use('Agg')

import PID
import time
import matplotlib.pyplot as plt
import numpy as np
from scipy.interpolate import spline


P = 1.4
I = 1
D = 0.001
pid = PID.PID(P, I, D)

pid.SetPoint = 0.0
pid.setSampleTime(0.01)

total_sampling = 100
feedback = 0

feedback_list = []
time_list = []
setpoint_list = []

print("simulating....")
for i in range(1, total_sampling):
    pid.update(feedback)
    output = pid.output
    if pid.SetPoint > 0:
        feedback += (output - (1 / i))

    if 20 < i < 60:
        pid.SetPoint = 1
```

```
    if 60 <= i < 80:
        pid.SetPoint = 0.5

    if i >= 80:
        pid.SetPoint = 1.3

    time.sleep(0.02)

    feedback_list.append(feedback)
    setpoint list.append(pid.SetPoint)
    time_list.append(i)

time_sm = np.array(time_list)
time_smooth = np.linspace(time_sm.min(), time_sm.max(), 300)
feedback_smooth = spline(time_list, feedback_list, time_smooth)

fig1 = plt.gcf()
fig1.subplots adjust(bottom=0.15)

plt.plot(time_smooth, feedback_smooth, color='red')
plt.plot(time_list, setpoint_list, color='blue')
plt.xlim((0, total_sampling))
plt.ylim((min(feedback_list) - 0.5, max(feedback_list) + 0.5))
plt.xlabel('time (s)')
plt.ylabel('PID (PV)')
plt.title('TEST PID')


plt.grid(True)
print("saving...")
fig1.savefig('result.png', dpi=100)
```

将该程序保存为 test_pid.py，然后运行该程序。

```
$ python test_pid.py
```

该程序将会生成 result.png 作为 PID 进程的结果。如图 1-35 所示为一个输出样例 result.png，可以看到深色线代表期望数值，浅色线是 PID 的输出结果。

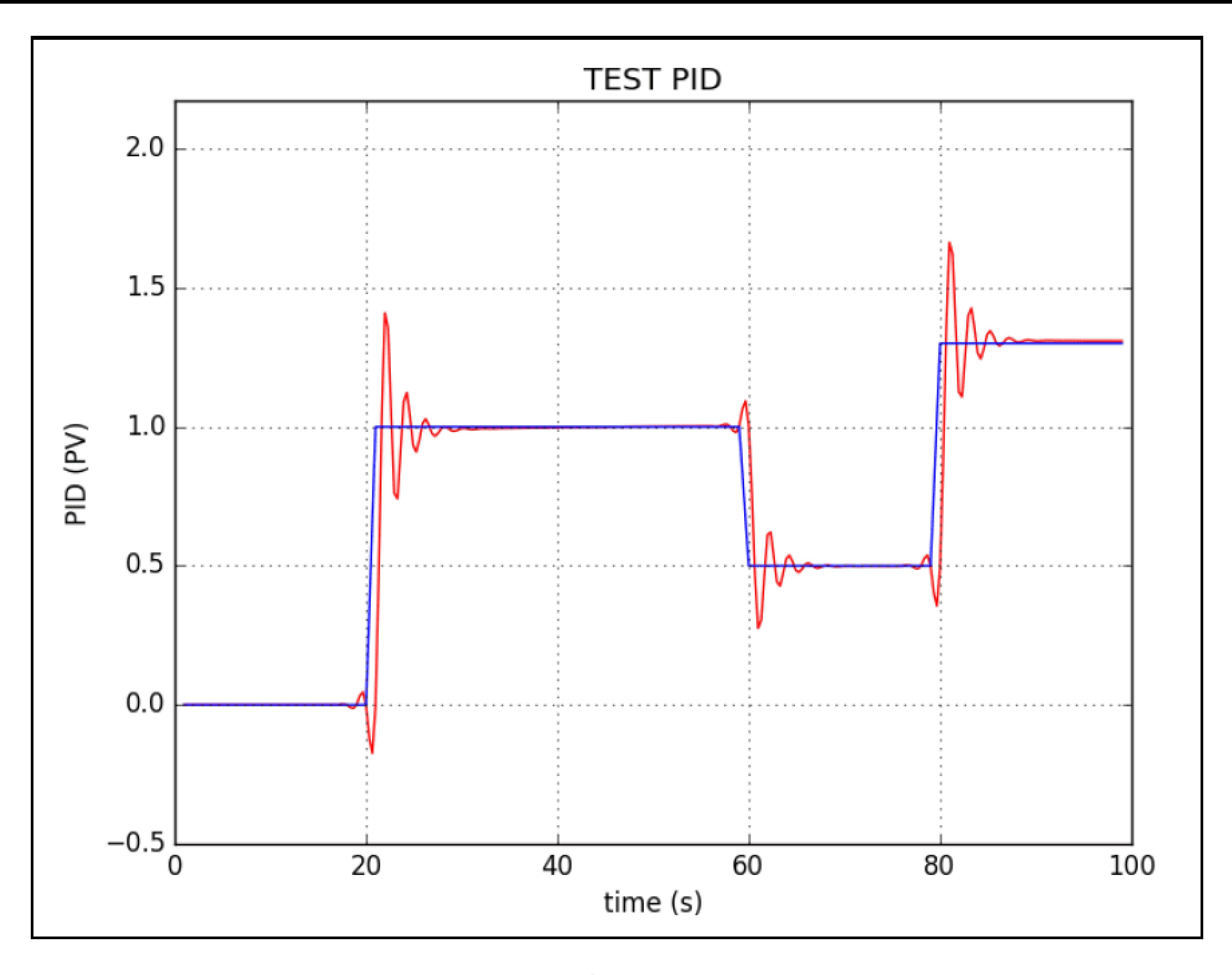


图 1-35

它是如何生效的？

首先，按如下方式定义自己的 PID 参数：

```
P = 1.4
I = 1
D = 0.001
pid = PID.PID(P, I, D)

pid.SetPoint = 0.0
pid.setSampleTime(0.01)

total sampling = 100
feedback = 0

feedback list = []
time_list = []
setpoint_list = []
```

这之后，在取样之时计算 PID 的数值。在本样例中，按如下方式设置期望输出值：

- ❑ 对于从 20 到 60 的取样，期望输出 1。

- ❑ 对于从 60 到 80 的取样，期望输出 0.5。
- ❑ 超过 80 的取样，期望输出 1.3。

```
for i in range(1, total_sampling):
    pid.update(feedback)
    output = pid.output
    if pid.SetPoint > 0:
        feedback += (output - (1 / i))

    if 20 < i < 60:
        pid.SetPoint = 1

    if 60 <= i < 80:
        pid.SetPoint = 0.5

    if i >= 80:
        pid.SetPoint = 1.3

    time.sleep(0.02)

    feedback_list.append(feedback)
    setpoint_list.append(pid.SetPoint)
time_list.append(i)
```

最后一步是生成一个报告并且保存为文件，将该文件命名为 result.png。

```
time_sm = np.array(time_list)
time_smooth = np.linspace(time_sm.min(), time_sm.max(), 300)
feedback_smooth = spline(time_list, feedback_list, time_smooth)

fig1 = plt.gcf()
fig1.subplots_adjust(bottom=0.15)

plt.plot(time_smooth, feedback_smooth, color='red')
plt.plot(time_list, setpoint_list, color='blue')
plt.xlim((0, total_sampling))
plt.ylim((min(feedback_list) - 0.5, max(feedback_list) + 0.5))
plt.xlabel('time (s)')
plt.ylabel('PID (PV)')
plt.title('TEST PID')
```

```
plt.grid(True)
print("saving...")
fig1.savefig('result.png', dpi=100)
```

使用 PID 控制器控制房间温度

现在可以使用真正的应用来改变 PID 控制器。使用 DHT22 来检查一个房间的温度。温度测量的输出被用来作为一个给 PID 控制器的反馈输入。

如果该 PID 输出正值，那么就开启加热器。否则，就激活制冷装置。这可能不是最好的方案，但这是个很好的出发点，可以让用户看到 PID 控制器是如何工作生效的。

把 DHT22 添加到 GPIO23（BCM）上。写上如下程序代码：

```
import matplotlib
matplotlib.use('Agg')

import PID
import Adafruit_DHT
import time
import matplotlib.pyplot as plt
import numpy as np
from scipy.interpolate import spline

sensor = Adafruit_DHT.DHT22

# DHT22 pin on Raspberry Pi
pin = 23

P = 1.4
I = 1
D = 0.001
pid = PID.PID(P, I, D)

pid.SetPoint = 0.0
pid.setSampleTime(0.25)  # a second

total_sampling = 100
sampling_i = 0
measurement = 0
```

```
feedback = 0

feedback list = []
time list = []
setpoint list = []

print('PID controller is running..')
try:
    while 1:
        pid.update(feedback)
        output = pid.output

        humidity, temperature = Adafruit DHT.read retry(sensor, pin)
        if humidity is not None and temperature is not None:
            if pid.SetPoint > 0:
                feedback += temperature + output

            print('i={0} desired.temp={1:0.1f}*C temp={2:0.1f}*C pid.out=
{3:0.1f} feedback={4:0.1f}'
                  .format(sampling_i, pid.SetPoint, temperature, output,
feedback))
            if output > 0:
                print('turn on heater')
            elif output < 0:
                print('turn on cooler')

        if 20 < sampling_i < 60:
            pid.SetPoint = 28  # celsius

        if 60 <= sampling_i < 80:
            pid.SetPoint = 25  # celsius

        if sampling_i >= 80:
            pid.SetPoint = 20  # celsius

        time.sleep(0.5)
        sampling_i += 1

        feedback list.append(feedback)
        setpoint_list.append(pid.SetPoint)
```

```
        time list.append(sampling i)

        if sampling i >= total sampling:
            break

except KeyboardInterrupt:
    print("exit")


print("pid controller done.")
print("generating a report...")
time_sm = np.array(time_list)
time_smooth = np.linspace(time_sm.min(), time_sm.max(), 300)
feedback_smooth = spline(time_list, feedback_list, time_smooth)

fig1 = plt.gcf()
fig1.subplots_adjust(bottom=0.15, left=0.1)

plt.plot(time smooth, feedback smooth, color='red')
plt.plot(time list, setpoint list, color='blue')
plt.xlim((0, total_sampling))
plt.ylim((min(feedback_list) - 0.5, max(feedback_list) + 0.5))
plt.xlabel('time (s)')
plt.ylabel('PID (PV)')
plt.title('Temperature PID Controller')


plt.grid(True)
fig1.savefig('pid temperature.png', dpi=100)
print("finish")
```

将该程序保存为文件并且命名为 ch01_pid.py。现在可以运行该程序：

```
$ sudo python ch01_pid.py
```

执行该程序之后，应该可以获得一个名为 pid_temperature.png 的文件。该程序的一个输出样例可以在图 1-36 中看到。

如果不采取任何措施，既不打开制冷装置也不打开制热装置，那么将获得如图 1-37 所示的运行输出结果。

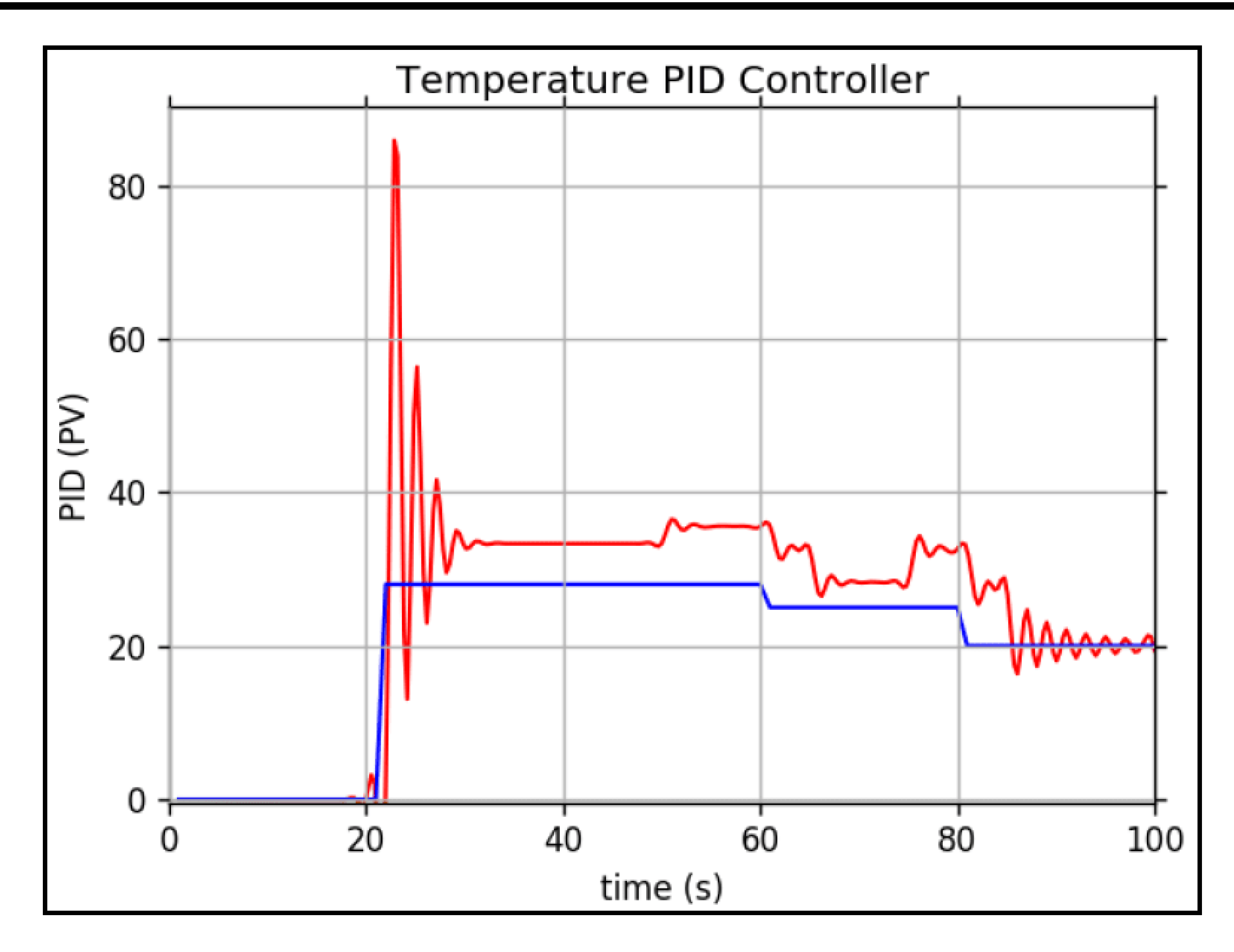


图 1-36

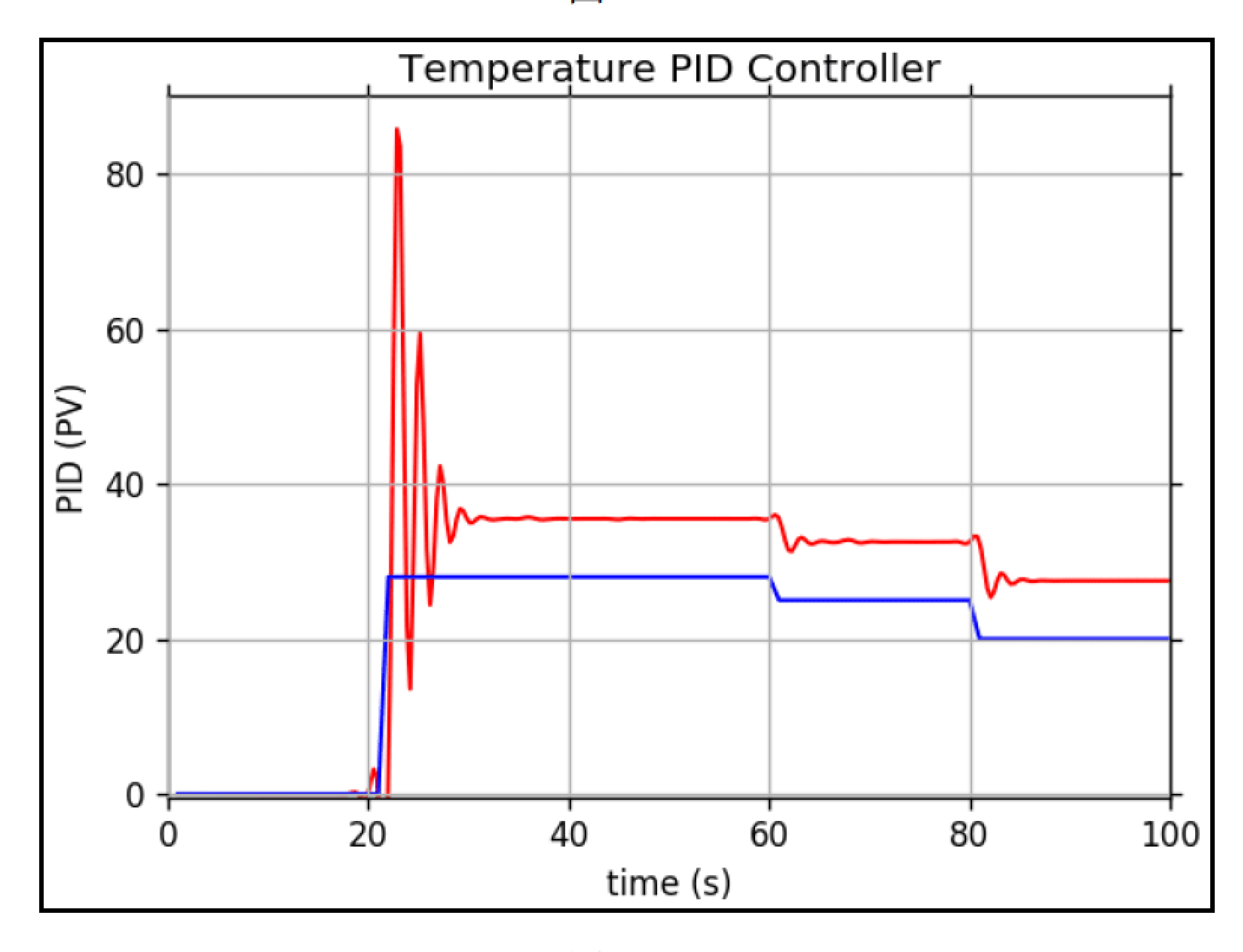


图 1-37

它是如何工作生效的？

大致来说，该程序结合了两个主题：通过 DHT22 读取当前温度数据和实现一个 PID 控制器。在测量数据之后，将该值发送到 PID 控制器程序。PID 的输出将会采取一个特定行动。在本次样例中，会打开制冷和制热装置。

总　　结

本章回顾了一些基本的统计学知识，并且探索了各种跟统计学和数据科学有关的 Python 函数库，同时也了解了几个物联网设备平台和如何感知与执行。

对于最后一个主题，配置了一个 PID 控制器作为一个学习样例来学习如何将控制器系统集成到一个物联网工程当中。在接下来的章节中，将会学习如何为物联网工程搭建一个决策系统。

引　　用

下面是一个推荐读书列表，从下列书籍中可以学习到更多的关于本章内容的主题。

1. Richard D. De Veaux, Paul F. Velleman, and David E. Bock, *Stats Data and Models*, 4th Edition, 2015, *Pearson Publishing*.

2. Sheldon M. Ross, *Introductory Statistics*, 3rd Edition, Academic Press, 2010.

第 2 章　将决策系统用于物联网工程

如果感觉着凉了，那么人们会穿上夹克。当饿了的时候，人们会决定吃饭。这些决定可以被很自然地做出，但是机器如何才能做出决定呢？本章将会学习如何搭建一个可以在物联网设备上使用的决策系统。

本章将探索如下主题：

- 决策系统和机器学习的基本介绍
- 探索搭建决策系统的 Python 函数库
- 搭建一个简单的基于贝叶斯定理的决策系统
- 结合决策系统和物联网工程
- 搭建基于决策系统的物联网

决策系统和机器学习基本介绍

决策系统是可以基于几个输入参数来做出决策的系统。决策系统搭建在决策定理的基础上。人们经常要为生活中的几乎任何事物做出决策。

下面是几个人们做出决策的示例：

我今天应该买这辆汽车吗？该决定取决于我的喜好。这辆车看起来很好，但是它对于我来说实在是太贵了。

我今天应该带一把雨伞吗？该决定取决于我们所在区域的当前状况。如果当前是阴天多云，最好还是带一把雨伞在身边，即使当天可能不会下雨。

大致说来，我们教会机器，比如说一台计算机，来理解并且实现一个目标。这被称为机器学习。很多不同的程序在机器上被实现，这样它们就可以做出决策。

机器学习提供了用来搭建决策系统的各种算法。在本书中，会使用模糊逻辑和贝叶斯算法来搭建一个决策系统。在接下来的一节中将对它们进行讲解。

用于决策系统的贝叶斯

贝叶斯算法使用条件概率的方法来解读数据。在本节中，使用贝叶斯模型来搭建一

个决策系统。

考虑字母 D，称之为决策空间，将用它来指代所有能被决策者 decision maker（DM）做出的决策 d 所组成的空间。Θ 代表所有可能的结果所组成的空间或者自然的状态 ω，$\omega \in \Theta$。

基于决策系统的贝叶斯由贝叶斯定理所构建。为了演示，将会使用贝叶斯原理构建一个简单的垃圾邮件过滤器。假设样本空间 X 是所有可能的单词组成的数据集，从该空间中可以生成任意单个数据集 x。对于任意 $\omega \in \Theta$ 和 $x \in X$，采样模块 $P(\omega)$描述了 x 是被垃圾邮件空间所生成的可能性。$P(x|\omega)$，先验分布，是真实的总体特征，并且假设了一个对于 x 的垃圾邮件概率。$P(\omega|x)$，后验分布，表述了已经观察了数据集 x 之后，ω 确实是来自垃圾邮件空间的可能性。

后验分布可通过如下贝叶斯定理来获得：

$$P(\omega \mid x) = \frac{P(x \mid \omega) P(\omega)}{P(x)}$$

这将会返回一个垃圾邮件的概率值。

现在可以搭建一个决策系统。考虑 $\lambda(\omega,d)$作为一个损失函数，它会告诉我们每个行动 d 带来的损失有多少。损失函数 $\lambda(d_i|\omega_i)$代表做出属于 ω_i 类别的行动 d_i 所会导致的损失。损失的期望或者条件风险被定义如下：

$$R(d_i|x) = \sum_{j=1}^{c} \lambda(d_i \mid \omega_j) P(\omega_j|x)$$

决策函数 $d(x)$是从观察到行动的映射函数。一个决策函数的总风险可以根据如下公式来进行计算：

$$E_{P(x)}[R(d(x) \mid x)] = \sum_{x} P(x) R(d(x) \mid x)$$

决策函数在它能最小化总风险时是最优的。决策将会以对于每个行动都选择最小风险的原则来做出。

这是一个简单的解释。为了能得到更多关于贝叶斯理论的信息，建议可以读一本关于贝叶斯模型的教科书。

用于决策系统的模糊逻辑

考虑想要基于当前温度做出一个决策。例如，如果房间的温度超过 30℃，那就要打开制冷装置。否则，如果房间的温度为 18℃，就要打开制热装置。

正因为已经定义了何时打开这些机器的具体数值，所以可以做出这些决策。如果房间的温度太高了，将会打开制冷装置。另外，当房间温度太低时，就想要打开制热装置。

冷和热是关于人类词汇的两个术语。我们应该定义什么是冷和热的标准。人类可以轻易地区别冷和热，但是怎么让计算机和其他机器知道呢？

这个问题可以用模糊逻辑来解决。模糊逻辑的思想首先在 20 世纪 60 年代被加州大学伯克利分校的 Dr. Lotfi Zadeh 所提出。模糊逻辑的理论定义于模糊集合和成员关系。

简而言之，基于决策系统的模糊逻辑如图 2-1 所示。

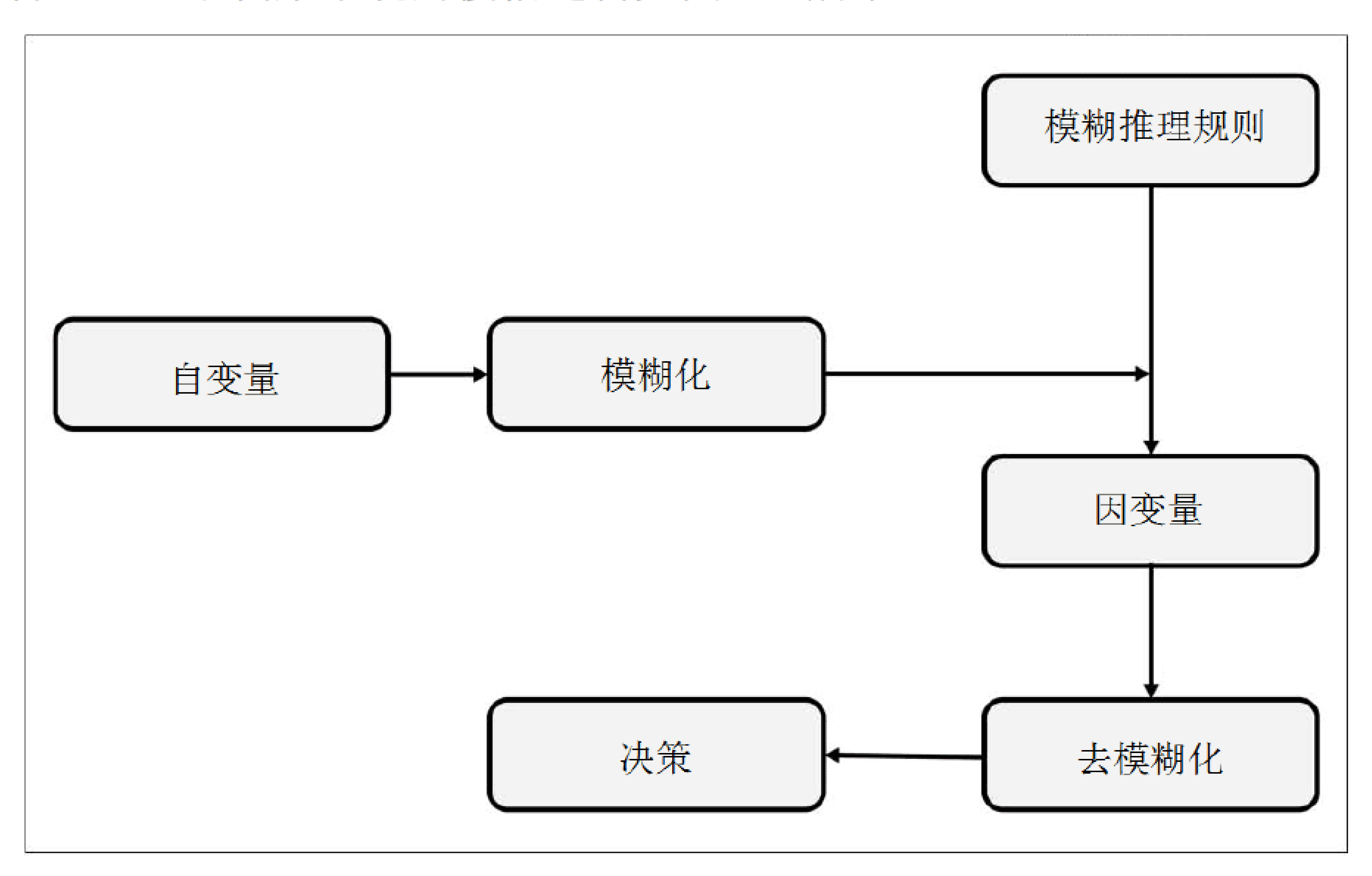


图 2-1

按照下列步骤搭建一个决策系统：

（1）定义代表你的问题的相互独立的变量。该步骤是抽取过程的一部分。这些变量通常有具体数值。

（2）搭建包含语言变量（比如说冷和热）的模糊集合。

（3）执行模糊化过程，该过程会将相互独立的数值变量转化为相互依赖的语言变量。

（4）建立模糊推理的规则以便于在给定输入和输出之间获得映射。可以使用 if-then 方式。

（5）在聚合了所有输出之后，进行去模糊化处理来获得一个单个的数值。

从单个数字的输出中，可以做出决策。在下一节中将用一个实验来展示如何使用模糊逻辑来搭建决策系统。

搭建决策系统所需的 Python 函数库

本节将探索一些 Python 函数库来搭建决策系统，将会重点讲用于实验决策系统的贝叶斯模型和模糊逻辑模型。

贝叶斯模型

可以用 Python 来实现贝叶斯概率。在本节的演示中，从两个独立变量 x_1 和 x_2 生成输出值。输出模型按如下方式定义：

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + c\sigma$$

c 是一个随机变量。将 α 、 β_1、 β_2 和 σ 分别定义为 0.5、1、2.5 和 0.5。

这些独立变量根据 NumPy 函数库中的随机对象而生成。这之后，用这些变量来计算模型。

可以使用下列脚本来实现该用例：

```
import matplotlib
matplotlib.use('Agg')

import numpy as np
import matplotlib.pyplot as plt

# 初始化
np.random.seed(100)
alpha, sigma = 0.5, 0.5
beta = [1, 2.5]
size = 100

# 预测变量
X1 = np.random.randn(size)
X2 = np.random.randn(size) * 0.37

# 模拟输出变量
Y = alpha + beta[0]*X1 + beta[1]*X2 + np.random.randn(size)*sigma

fig, ax = plt.subplots(1, 2, sharex=True, figsize=(10, 4))
fig.subplots_adjust(bottom=0.15, left=0.1)
```

```
ax[0].scatter(X1, Y)
ax[1].scatter(X2, Y)
ax[0].set_ylabel('Y')
ax[0].set_xlabel('X1')
ax[1].set_xlabel('X2')


plt.grid(True)
fig.savefig('predict.png', dpi=100)
print("finish")
```

可以将这些脚本保存为一个文件并将其命名为 ch02_predict.py。
接着，可以通过输入下列命令运行该程序：

```
$ python ch02_predict.py
```

该程序将会生成一幅 PNG 文件 predict.png，如图 2-2 所示。

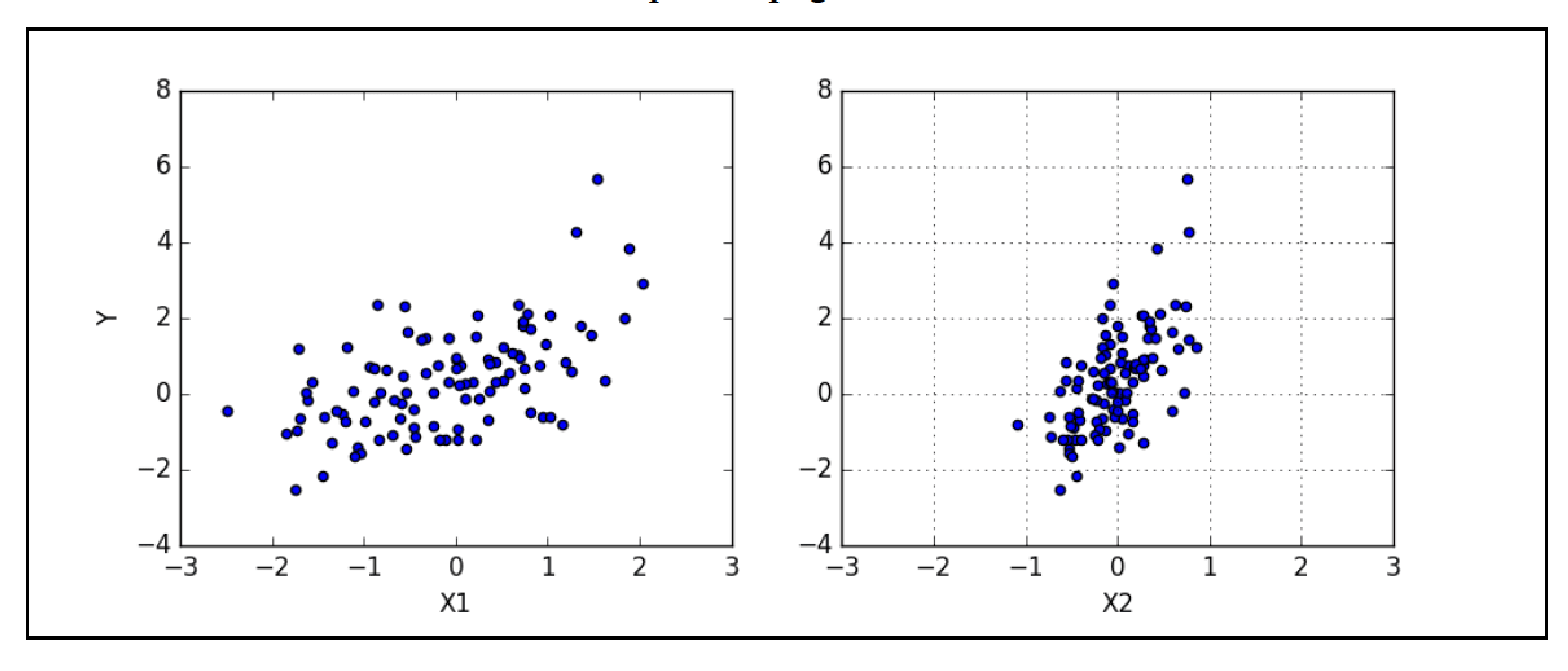


图 2-2

这个程序很简单，通过 NumPy 生成独立的变量。所以，下面将使用 matplotlib 函数库画出该模型。

现在尝试着用 Python 函数库来搭建一个贝叶斯模型。Python 的其中一个用于贝叶斯计算的函数库就是 PyMC，它提供贝叶斯统计模型和拟合算法，包括马尔科夫链蒙特卡洛算法。PyMC 是一个开源的函数库，可以通过网址 https://github.com/pymc-devs/pymc 来查看和下载。

为了安装 PyMC 函数库，可以使用 easy_install 或者 pip。如果想要使用 easy_install 安装 PyMC，可以输入下列命令：

```
$ easy_install pymc
```

用户可能会需要管理员级别的权限（sudo）来通过 easy_install 安装 PyMC。另外一种选择是通过 pip 来安装 PyMC：

```
$ pip install l pymc
```

为了测试 PyMC，使用来自 PyMC 的示例代码。实现贝叶斯计算的第一步就是搭建一个模型。在这个情境下，使用正态分布作为先验分布的参数：

$$\theta(x) = \frac{e^{a+bx}}{(1+e^{a+bx})}$$

现在，可以使用 PyMC 来实现这个问题。请写出如下脚本：

```
import pymc
import numpy as np

# 一些数据
n = 5 * np.ones(4, dtype=int)
x = np.array([-.86, -.3, -.05, .73])

# 对于未知参数的先验分布
alpha = pymc.Normal('alpha', mu=0, tau=.01)
beta = pymc.Normal('beta', mu=0, tau=.01)

# 任意特定的参数函数
@pymc.deterministic
def theta(a=alpha, b=beta):
    """theta = logit^{-1}(a+b)"""
    return pymc.invlogit(a + b * x)
# 数据的二项分布可能性
d = pymc.Binomial('d', n=n, p=theta, value=np.array([0., 1., 3., 5.]),
                      observed=True)
```

将该模型保存为文件并且命名为 mymodel.py。

我们的模型将通过一种模拟方法被使用，该模拟方法被称为马尔科夫链蒙特卡洛（MCMC）方法。将从该模拟方法中获得后验分布数值。

创建一个文件，将该文件命名为 ch02_pymc.py 并且写出如下脚本：

```
import matplotlib
matplotlib.use('Agg')

import pymc
import mymodel

S = pymc.MCMC(mymodel, db='pickle')
S.sample(iter=10000, burn=5000, thin=2)

pymc.Matplot.plot(S)
print("finish")
```

可以通过输入如下命令在 Raspberry Pi Terminal 上运行该程序：

```
$ python ch02_pymc.py
```

如果该程序运行良好，可以看到如图 2-3 所示的输出画面。

```
Documents — pi@raspberrypi: ~/Documents/book — ssh pi@192.168.0.12 — 8...
pi@raspberrypi:~/Documents/book $ python -W ignore ch02_pymc.py
[----             10%                  ] 1077 of 10000 complete in 0.5 sec
[--------         21%                  ] 2155 of 10000 complete in 1.0 sec
[------------     32%                  ] 3232 of 10000 complete in 1.5 sec
[---------------- 43%                  ] 4309 of 10000 complete in 2.0 sec
[-----------------53%                  ] 5330 of 10000 complete in 2.5 sec
[-----------------62%---               ] 6252 of 10000 complete in 3.0 sec
[-----------------71%-------           ] 7172 of 10000 complete in 3.5 sec
[-----------------80%----------        ] 8092 of 10000 complete in 4.0 sec
[-----------------90%--------------    ] 9012 of 10000 complete in 4.5 sec
[-----------------99%----------------- ] 9932 of 10000 complete in 5.0 sec
[-----------------100%-----------------] 10000 of 10000 complete in 5.0 sec
Plotting theta_0
Plotting theta_1
Plotting theta_2
Plotting theta_3
Plotting beta
Plotting alpha
finish
pi@raspberrypi:~/Documents/book $
```

图 2-3

该程序同时也会生成 3 个文件，即 alpha.png、beta.png 和 theta-3.png。alpha.png 文件的一个样例如图 2-4 所示。

可以在 alpha.png 中看到 alpha 数值，这些数值是服从正态分布的随机变量。另外，beta 值也是由正态分布所生成。可以在图 2-5 所示的 beta.png 中看到 beta 的数值。

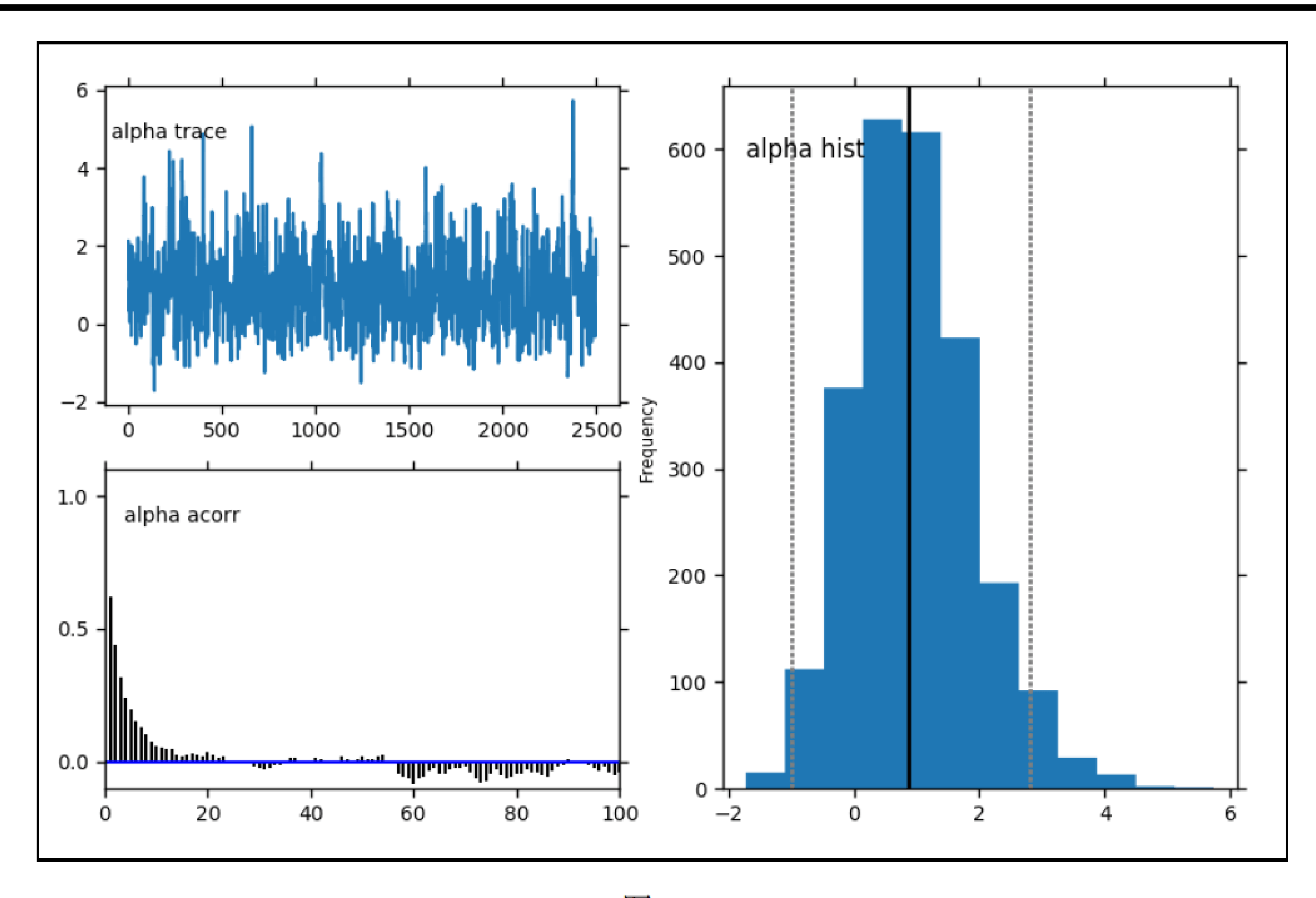


图 2-4

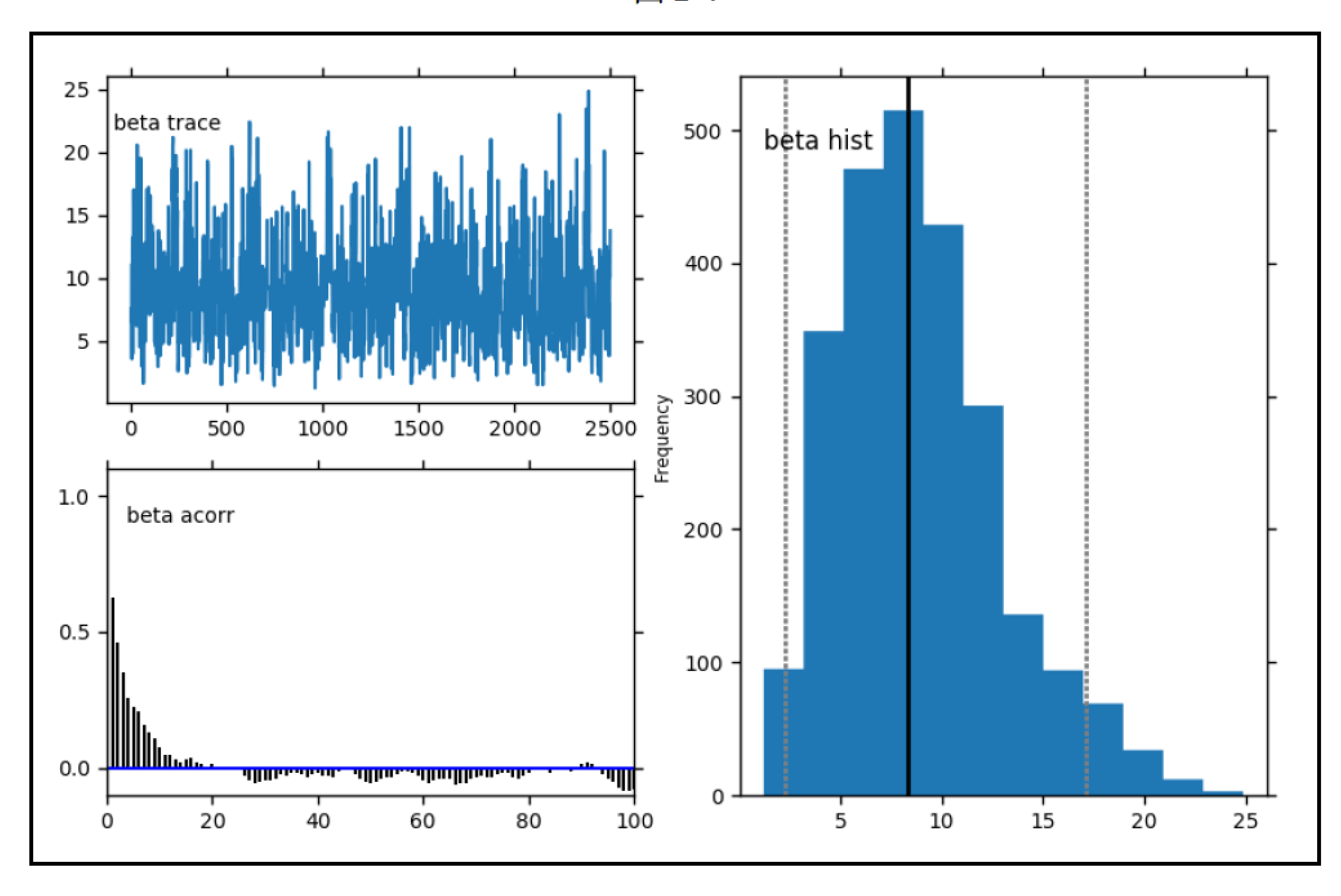


图 2-5

该程序的最后一张输出图是 theta-3.png 文件，这幅图显示了 theta 数值是如何通过一


个公式而被计算的，可以在图 2-6 中看到该结果。

图 2-6

模糊逻辑

Python 函数库中一个非常著名的模糊逻辑库是 scikit-fuzzy。已有几个模糊逻辑算法基于该函数库被实现。由于 scikit-fuzzy 是一个开源的函数库，可以通过网址 https://github.com/scikit-fuzzy/scikit-fuzzy 查看该函数库的源代码。

在安装该函数库之前，请确保已经安装了 NumPy 和 SciPy 函数库。可以通过 pip 来安装 scikit-fuzzy。请输入下列命令：

```
$ sudo pip install scikit-fuzzy
```

作为另一种选择，也可以从源代码来安装 scikit-fuzzy。请输入下列命令：

```
$ git clone https://github.com/scikit-fuzzy/scikit-fuzzy
$ cd scikit-fuzzy/
$ sudo python setup.py install
```

完成安装之后，就可以使用 scikit-fuzzy 了。

为了测试怎样通过 scikit-fuzzy 实现工作，将使用 fuzz.trimf()函数来为温度数据搭建一个模糊成员关系（membership）。可以写出如下脚本：

```
import matplotlib
matplotlib.use('Agg')


import numpy as np
import skfuzzy as fuzz
import matplotlib.pyplot as plt

# 生成 universe 变量
x_temp = np.arange(0, 11, 1)

# 生成模糊 membership 函数
temp_lo = fuzz.trimf(x_temp, [0, 0, 5])
temp_md = fuzz.trimf(x_temp, [0, 5, 10])
temp hi = fuzz.trimf(x temp, [5, 10, 10])

# 可视化 universes 和 membership 函数
fig, ax = plt.subplots()

ax.plot(x_temp, temp_lo, 'b--', linewidth=1.5, label='Cold')
ax.plot(x temp, temp md, 'g-', linewidth=1.5, label='Warm')
ax.plot(x_temp, temp_hi, 'r:', linewidth=1.5, label='Hot')
ax.set_title('Temperature')
ax.legend()

ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.get xaxis().tick bottom()
ax.get_yaxis().tick_left()
ax.set_ylabel('Fuzzy membership')

plt.tight_layout()

print('saving...')
plt.grid(True)
```

```
fig.savefig('fuzzy_membership.png', dpi=100)
print('done')
```

将这些脚本保存为文件并命名为 ch02_skfuzzy.py。

现在可以通过输入下列命令来运行该文件：

```
$ python ch02_skfuzzy.py
```

该程序将会生成一个 fuzzy_membership.png 文件。该文件的一个样例如图 2-7 所示。

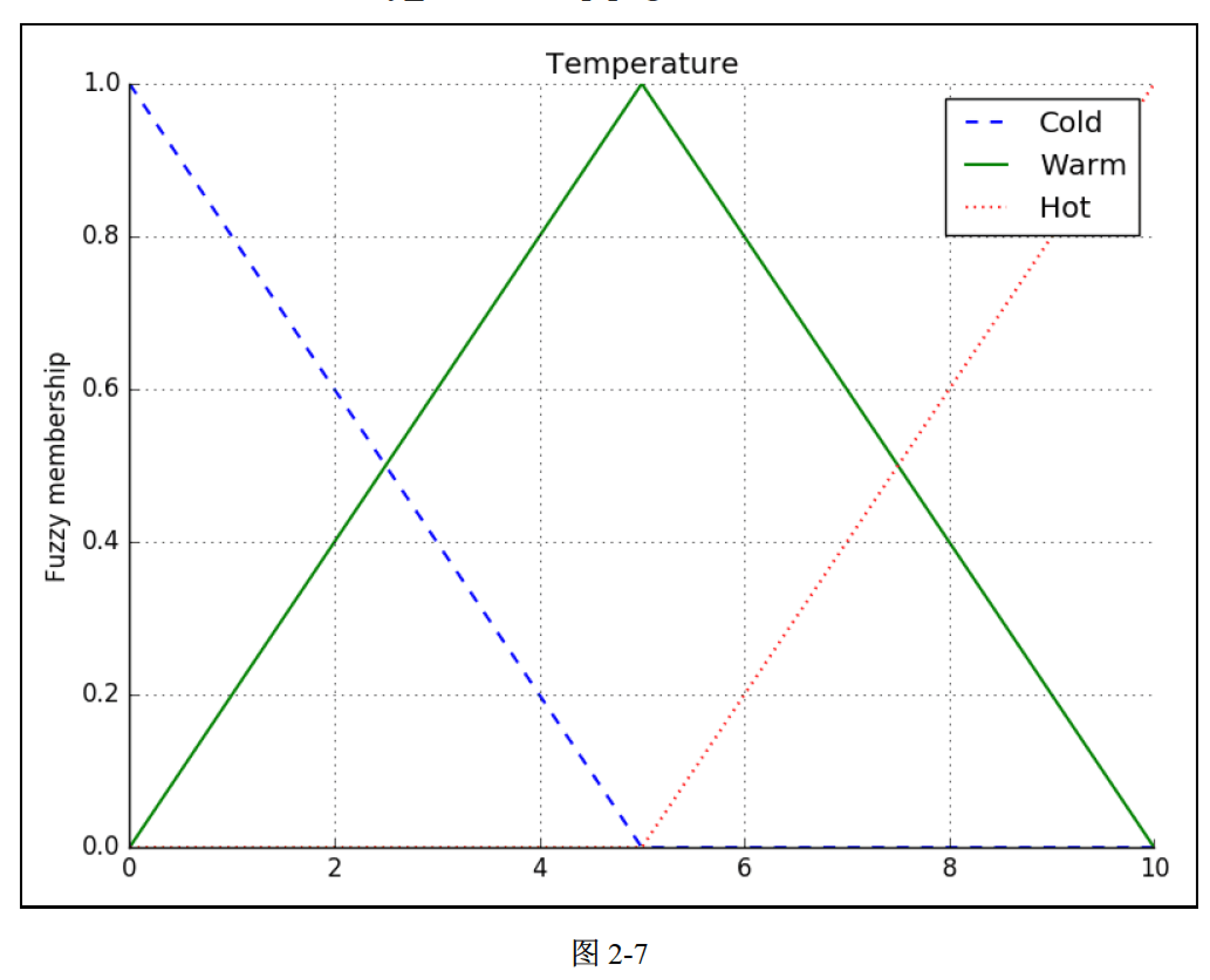


图 2-7

搭建一个简单的基于贝叶斯理论的决策系统

本节将会使用贝叶斯理论来搭建一个简单的决策系统。智能水系统是一个可以控制水的智能系统。大致说来，可以在图 2-8 中看到系统架构。

使用水检测处理来获得水源质量之后，就可以做出决策。如果水质比较好，就可以把水输送给顾客。否则，需要对水进行净化处理。

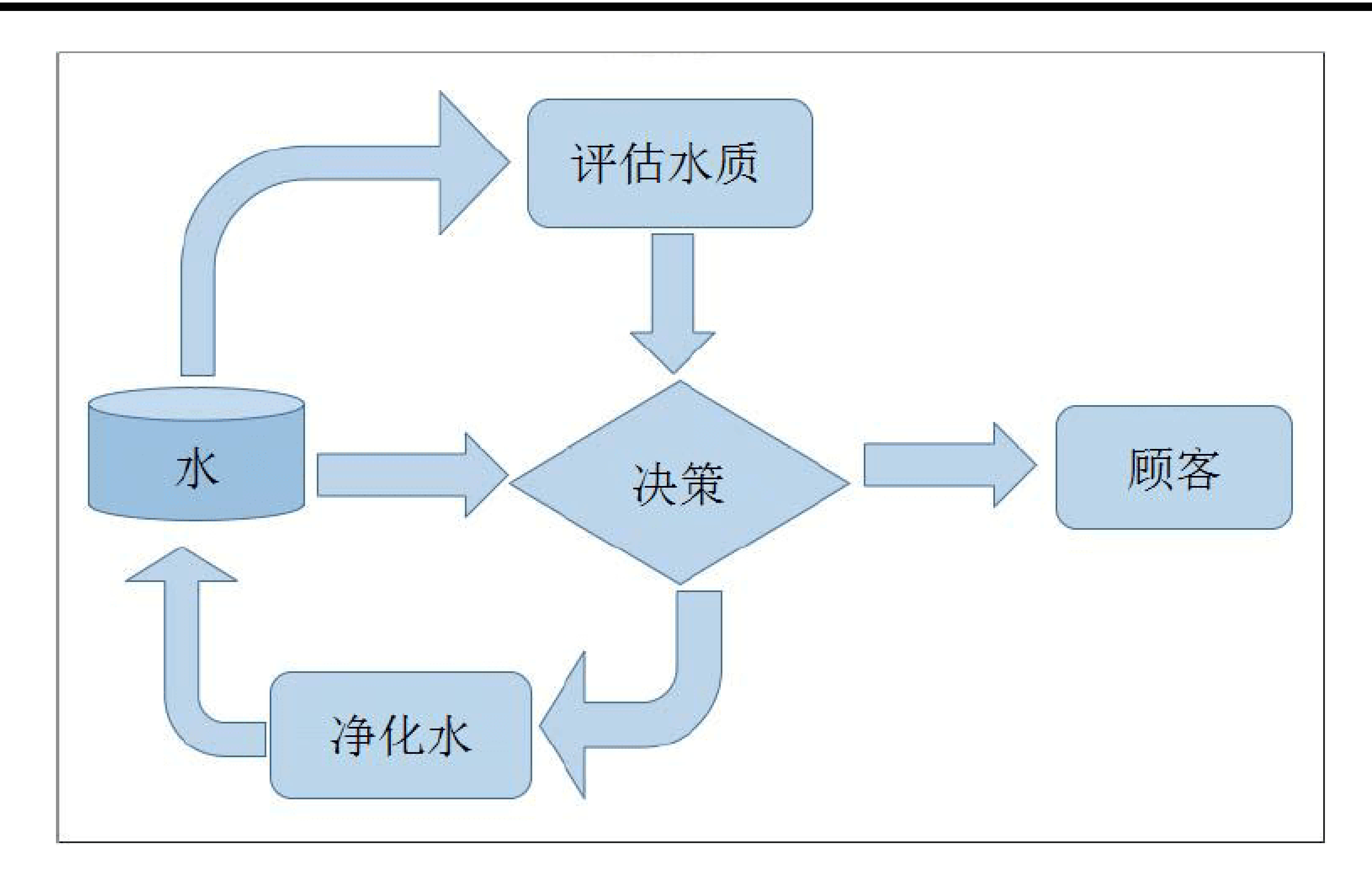


图 2-8

为了搭建基于贝叶斯理论的决策系统，首先需要定义自然状态。在该用例中，定义两种自然状态。

- ω_1：水可以被直接饮用。
- ω_2：水需要被净化处理（kotor）。

对于输入，可以分别声明 x_1 和 x_2 为负数和正数作为观测结果。

按如下方式定义先验分布数值和条件概率：

$$P(\omega_1) = 0.8$$

$$P(\omega_2) = 0.2$$

$$P(x_1 \mid \omega_1) = 0.3$$

$$P(x_1 \mid \omega_2) = 0.7$$

$$P(x_2 \mid \omega_1) = 0.2$$

$$P(x_2 \mid \omega_2) = 0.8$$

为了做出一个决策，应该定义一个损失函数。下面是一个对于程序所使用的损失函数：

$$\lambda(d_1 \mid \omega_1) = 0$$

$$\lambda(d_1 \mid \omega_2) = 5$$

$$\lambda(d_2 \mid \omega_1) = 10$$

$$\lambda(d_2 \mid \omega_2) = 0$$

现在可以写出该程序的完整脚本：

```
# 决策行动
# d1 = distribute water
```

```
# d2 = cleaning the water


# 先验
p_w1 = 0.8
p_w2 = 0.2

# 损失函数矩阵
lambda_1_1 = 0
lambda_1_2 = 5
lambda_2_1 = 10
lambda_2_2 = 0


# 每个类的条件概率可能性
# 基于水质是好是坏的观测值
# x1 = 负数
# x2 = 正数
p_x1_w1 = 0.3
p_x1_w2 = 0.7
p_x2_w1 = 0.2
p_x2_w2 = 0.8


# 计算 p_x1 和 p_x2
p_x1 = p_x1_w1 * p_w1 + p_x1_w2 * p_w2
p_x2 = p_x2_w1 * p_w1 + p_x2_w2 * p_w2


# 基于观测结果计算条件风险
p_w1_x1 = (p_x1_w1 * p_w1) / p_x1
p_w2_x1 = (p_x1_w2 * p_w2) / p_x1
p_w1_x2 = (p_x2_w1 * p_w1) / p_x2
p_w2_x2 = (p_x2_w2 * p_w2) / p_x2


r_d1_x1 = p_w1_x1 * lambda_1_1 + p_w2_x1 * lambda_1_2
r_d2_x1 = p_w1_x1 * lambda_2_1 + p_w2_x1 * lambda_2_2
r_d1_x2 = p_w1_x2 * lambda_1_1 + p_w2_x2 * lambda_1_2
r_d2_x2 = p_w1_x2 * lambda_2_1 + p_w2_x2 * lambda_2_2
```

```
print("r_a1_x1: ", r_d1_x1)
print("r_a2_x1: ", r_d2_x1)
print("r_a1_x2: ", r_d1_x2)
print("r_a2_x2: ", r_d2_x2)


# 计算总损失函数
e_d1 = p_x1 * r_d1_x1 + p_x2 * r_d1_x2
e_d2 = p_x1 * r_d2_x1 + p_x2 * r_d2_x2
print("e_d1: ", e_d1)
print("e_d2: ", e_d2)


if e_d1 < e_d2:
    print("final decision: d1 - distribute water")
else:
    print("final decision: d2 - cleaning the water")
```

将该段程序保存为文件并且命名为 ch02_bayes_theory.py。接着。通过运行下列命令来执行这段程序：

```
$ python ch02_bayes_theory.py
```

可以在图 2-9 中看到程序输出的一个示例。

```
Documents — pi@raspberrypi: ~/Documents/book — ssh pi@192.168.0.12 — 8...
pi@raspberrypi:~/Documents/book $ python ch02_bayes_theory.py
('r_a1_x1: ', 1.8421052631578947)
('r_a2_x1: ', 6.315789473684211)
('r_a1_x2: ', 2.5)
('r_a2_x2: ', 5.0)
('e_d1: ', 1.5)
('e_d2: ', 4.0)
final decision: d1 - distribute water
pi@raspberrypi:~/Documents/book $
```

图 2-9

可以通过改变先验分布和每个类的条件概率数值来做更多实验。

将决策系统和物联网项目结合

物联网开发板帮助人们执行检测和启动任务。为了在物联网开发板上搭建决策系统，

可以在物联网开发板上使用一个检测进程作为对决策系统的输入参数。在执行完决策计算之后，可以通过启动物联网开发板来做出一些动作。

总而言之，可以把决策系统结合到物联网开发板中，如图 2-10 所示。

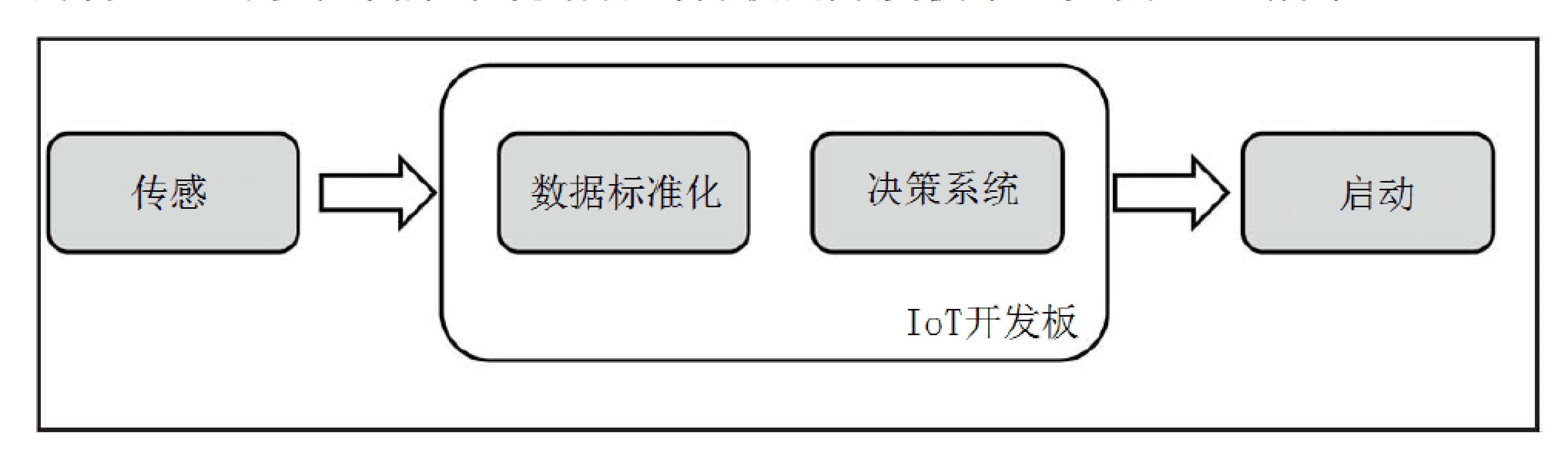
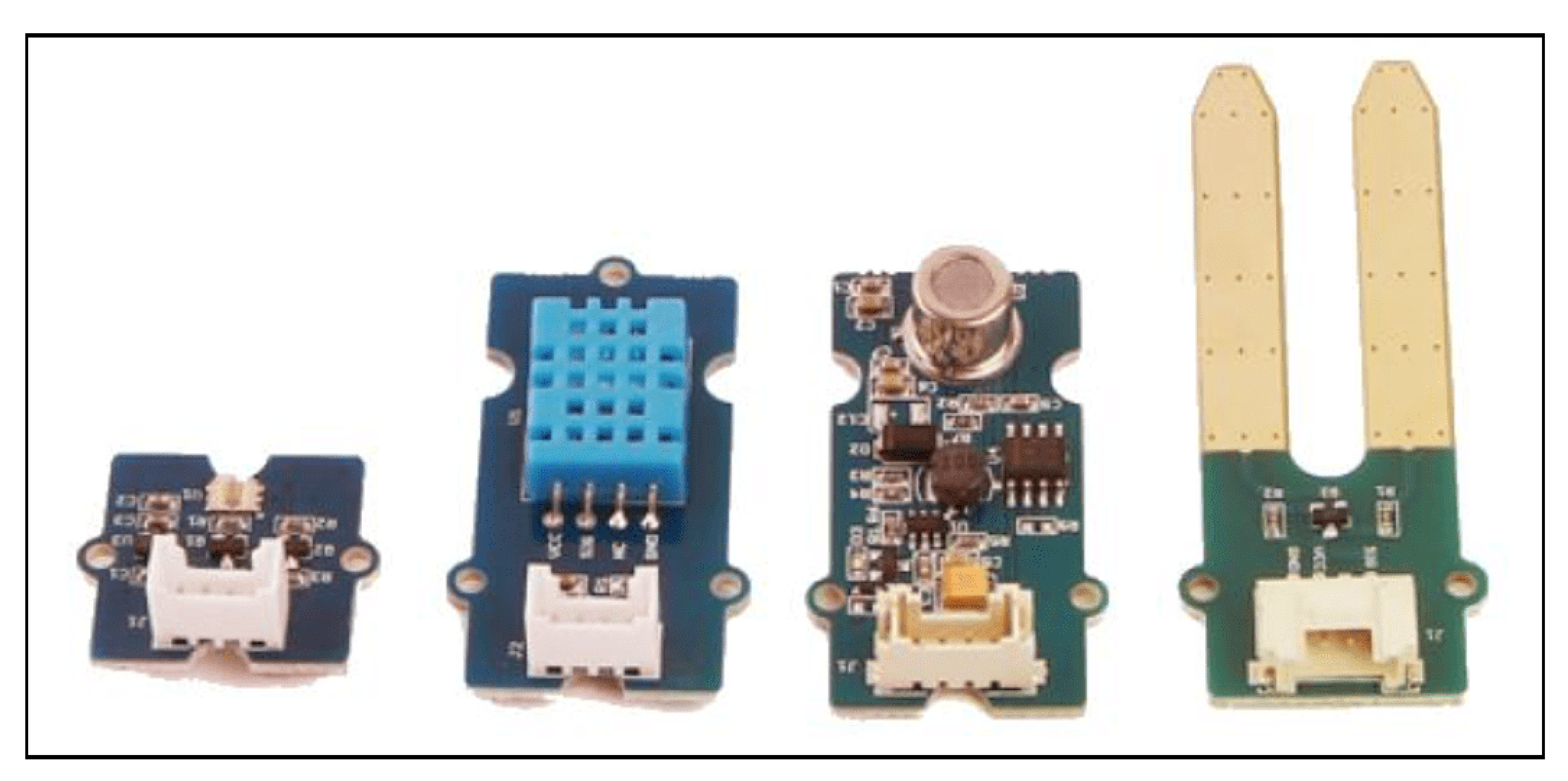


图 2-10

检测装置可以被添加到用于检测的物联网开发板上。根据需要，可以搜集环境数据，如温度，就像要被用到决策系统上的电子输入一样。可以在图 2-11 中看到检测设备的样例。



来源：http://www.seeedstudio.com

图 2-11

有多种执行设备可以被使用在决策系统中。来自一个系统的每种最终输出都可以被映射到一个具体行动中，该行动可以被表示为开启执行设备。

一些系统也许不会在它们的环境中做检测来搜集输入数据，可以从外部数据库或者通过网络从另外一个系统中来获得所需要的数据。

搭建基于决策系统的物联网

本节将在 Raspberry Pi 上通过模糊逻辑来搭建一个简单的决策系统，使用 Python 来实现该系统。

下面搭建一个系统来检测房间中的温度和湿度以便于判断该房间是否舒适。如果该环境不够舒适，那么就开启制冷装置。

图 2-12 显示了设计结构。

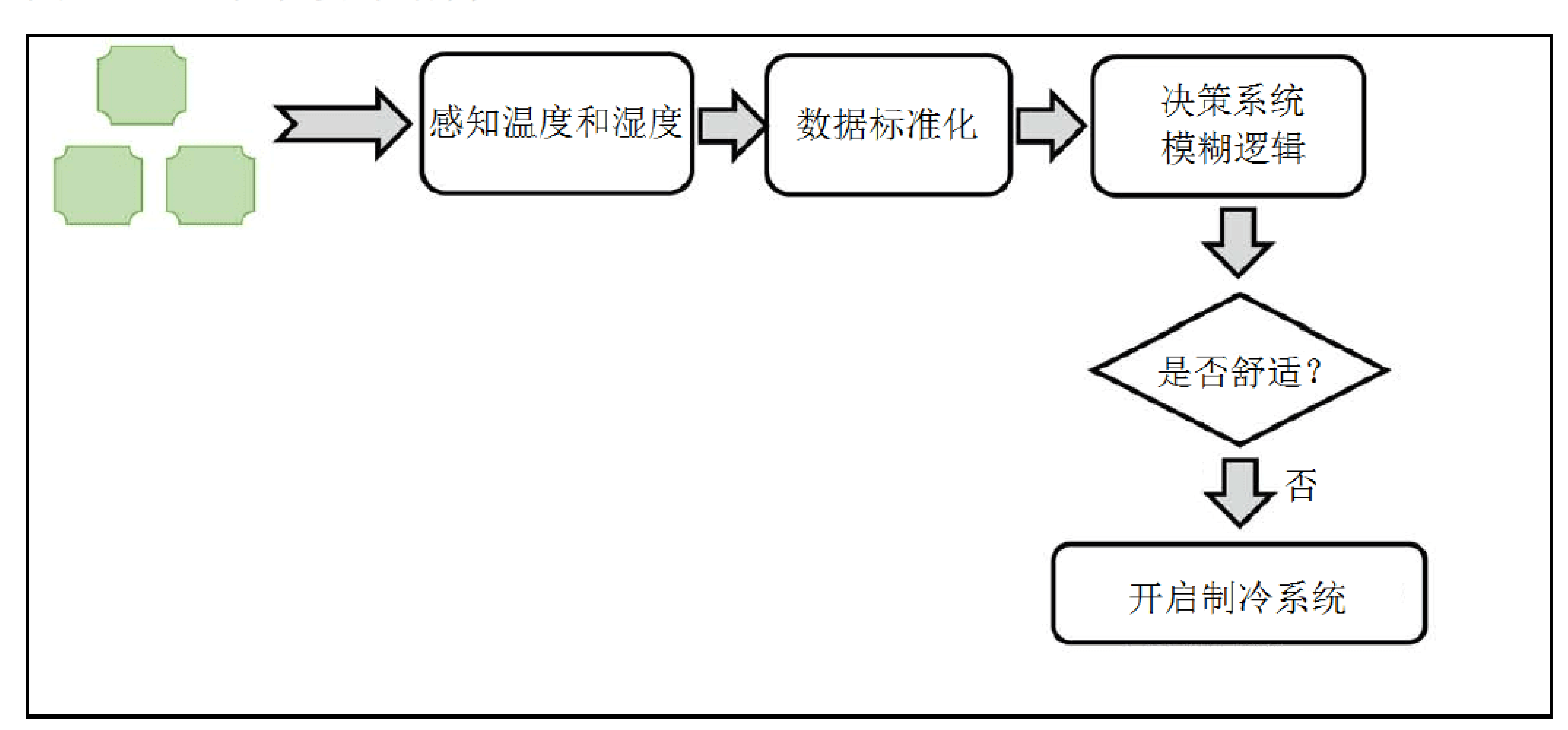


图 2-12

为了检测温度和湿度，使用 DHT22 模块。在第 1 章中已学习了这个模块。一个继电器模块被用来连接 Raspberry Pi 和一个制冷装置。

让我们开始搭建我们的系统吧。

布线

使用 DHT22 和继电器模块用于布线。把 DHT22 模块和下列连接处连接：

- DHT22 pin 1（VDD）连接至 Raspberry Pi 的 3.3V pin。
- DHT22 pin 2（SIG）连接至 Raspberry Pi 的 GPIO23（见 BCM 列）。
- DHT22 pin 4（GND）连接至 Raspberry Pi 的 GND pin。
- 一个继电器 VCC 连接至 Raspberry Pi 的 3.3V pin。
- 一个继电器 GND 连接至 Raspberry Pi 的 GND pin。

- 一个继电器信号连接至 Raspberry Pi 的 GPIO26（见 BCM 列）。

完整的布线结构如图 2-13 所示。

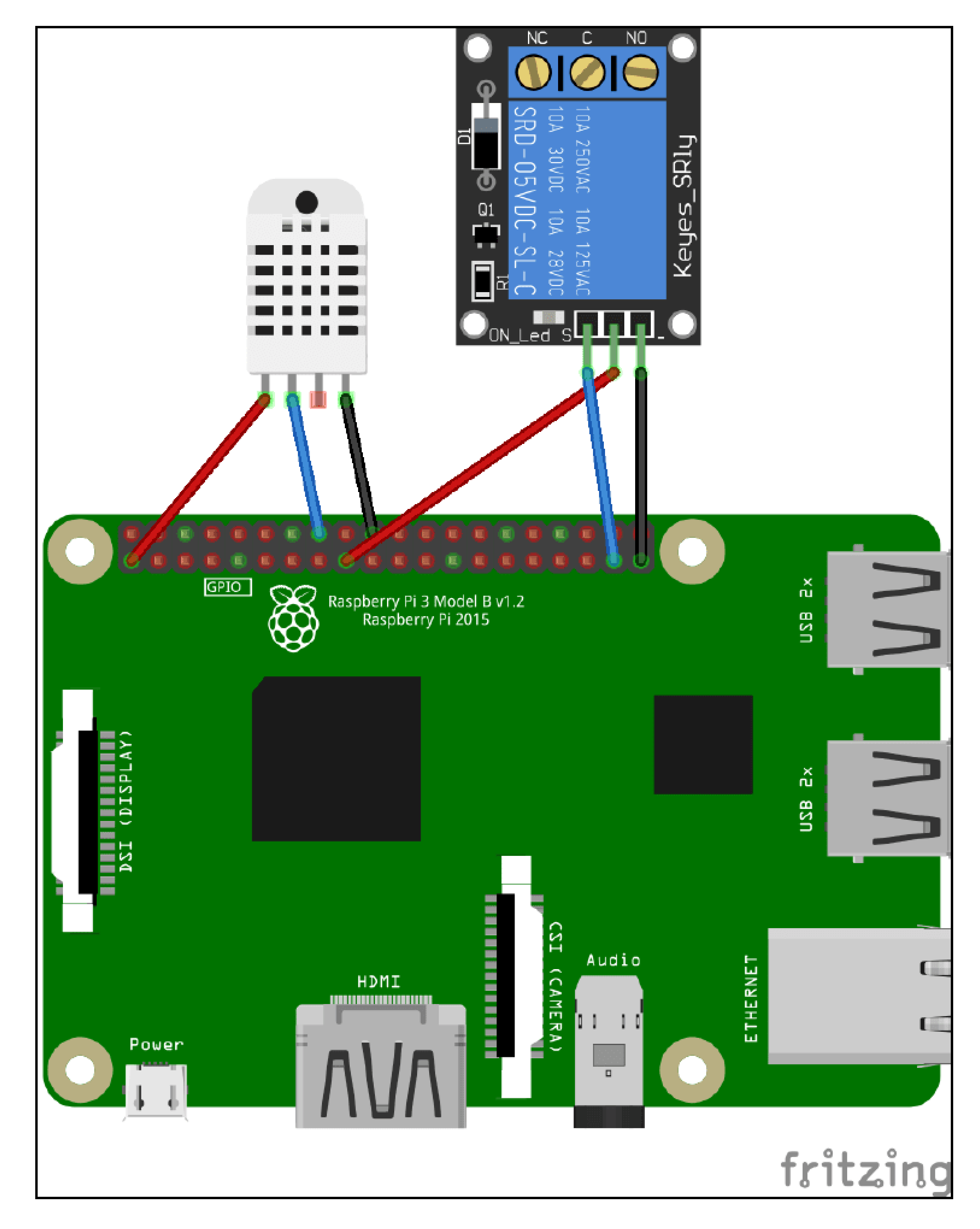


图 2-13

编写 Python 程序

我们建立了一段模糊逻辑来实现一个决策系统。来自检测器的两个输入是温度和湿度。在这个情形下，开始为温度和湿度设计模糊 membership。

为了进行测试，为温度和湿度编写了如图 2-14 所示的模糊 membership 模块。

在温度模块中，创建了 3 个种类：冷、温暖和热。另外，为湿度设计了两个种类：低湿度和高湿度。

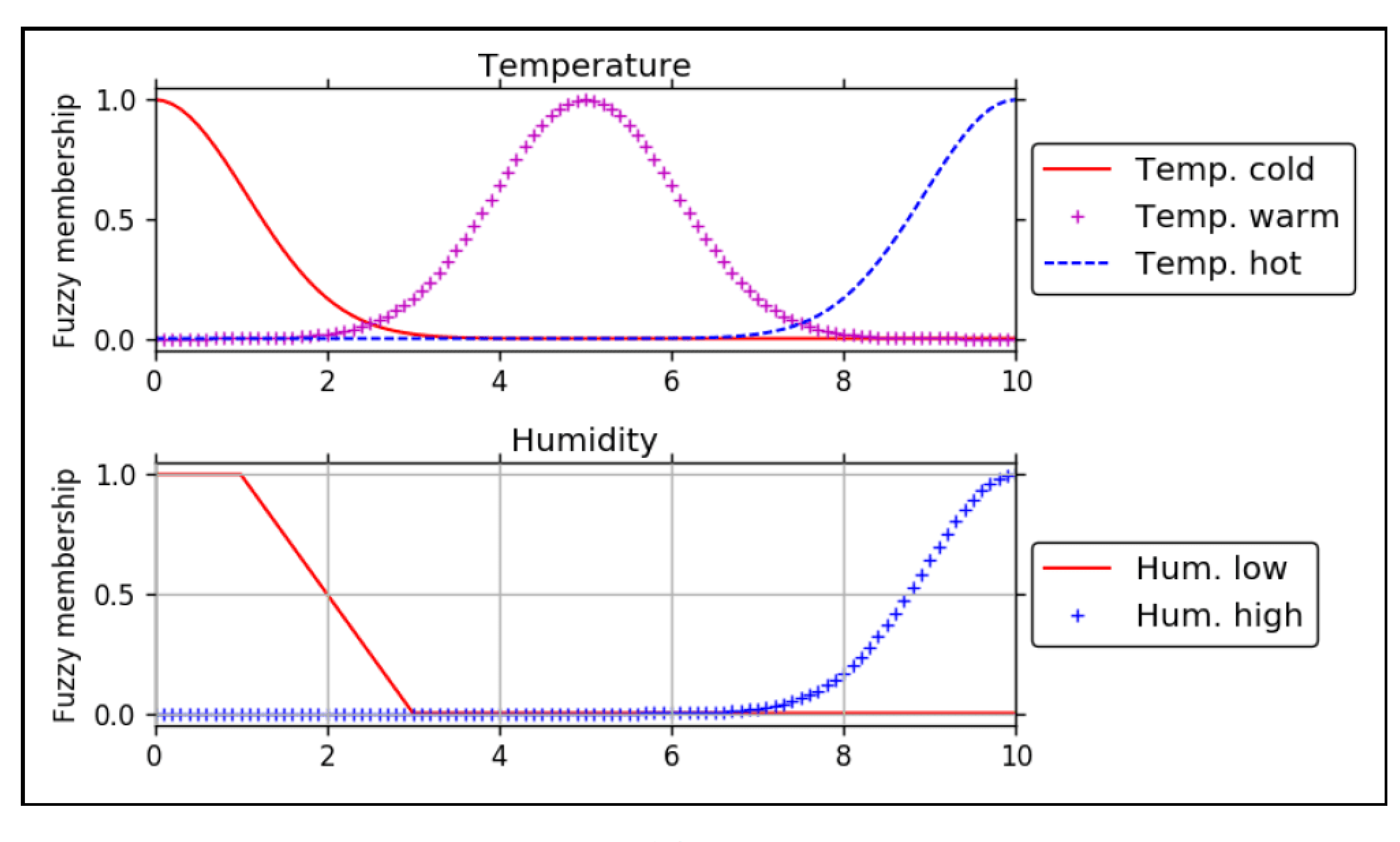


图 2-14

现在开始编写该程序。首先，创建一个文件，命名为 ch02_fuzzy.py。接着，初始化所需要的 Python 函数库：

```
import matplotlib
matplotlib.use('Agg')

import numpy as np
import skfuzzy as fuzz
import matplotlib.pyplot as plt

import Adafruit_DHT
import RPi.GPIO as GPIO
import time
```

这之后，为 DHT22 和继电器模块初始化 Raspberry Pi GPIO。

```
print('initialization...')

### 初始化 GPIO
relay_pin = 26
GPIO.setmode(GPIO.BCM)
GPIO.setup(relay_pin, GPIO.OUT)

sensor = Adafruit_DHT.DHT22
```

```
# Raspberry Pi 上的 DHT22 pin
dht_pin = 23
```

下一步就是通过为温度和湿度开始创建模糊 membership 来建立模糊逻辑。

创建 temperature_category()和 humidity_category()函数来将检测输入映射到系统。

```
########## 输入 #######################
#输入 universe 函数
temperature = np.arange(0, 11, 0.1)
humidity = np.arange(0, 11, 0.1)

# 输入 membership 函数
# 温度
temperature_cold = fuzz.gaussmf(temperature, 0, 1.5)
temperature_warm = fuzz.gaussmf(temperature, 5, 1.5)
temperature_hot = fuzz.gaussmf(temperature, 10, 1.5)
# 湿度
humidity_low = fuzz.trapmf(humidity, [0, 0, 1, 3])
humidity high = fuzz.gaussmf(humidity, 10, 1.5)

########## 输出 #######################
# comfort
#输出变量领域
comfort = np.arange(0, 30, 0.1)
# 输出 membership 函数
comfort_low = fuzz.trimf(comfort, [0, 5, 10])
comfort_ave = fuzz.trimf(comfort, [10, 15, 25])
comfort_very = fuzz.trimf(comfort, [20, 25, 30])


def temperature_category(temperature_in=18):
    temperature_cat_cold = fuzz.interp_membership(temperature,
temperature cold, temperature in)
    temperature cat warm = fuzz.interp membership(temperature,
temperature warm, temperature in)
    temperature cat hot = fuzz.interp membership(temperature,
temperature hot, temperature in)
    return dict(cold=temperature_cat_cold, warm=temperature_cat_warm,
hot=temperature_cat_hot)


def humidity_category(humidity_in=2):
    humidity_cat_low = fuzz.interp_membership(humidity, humidity_low,
```

```
humidity_in)
    humidity_cat_high = fuzz.interp_membership(humidity, humidity_
high, humidity_in)
    return dict(low=humidity_cat_low, high=humidity_cat_high)
```

也将 membership 作为应用输出到一个文件，这可以通过使用一个 matplotlib 函数库来完成。为温度和湿度保存模糊 memberships。

```
# 打印 membership
# 可视化 universes 和 membership 函数
print('saving membership...')
fig, ax = plt.subplots(2, 1)

[t1, t2, t3] = ax[0].plot(temperature, temperature_cold, 'r',
temperature, temperature_warm, 'm+', temperature,
                          temperature_hot, 'b--', label=['Temp. cold',
'Temp. warm', 'Temp. hot'])
ax[0].set_ylabel('Fuzzy membership')
ax[0].set title('Temperature')
ax[0].set ylim(-0.05, 1.05)
ax[0].set xlim(0, 10)

lgd1 = ax[0].legend([t1, t2, t3], ['Temp. cold', 'Temp. warm', 'Temp.
hot'], loc='center left', bbox_to_anchor=(1, 0.5))


[t1, t2] = ax[1].plot(humidity, humidity_low, 'r', humidity, humidity_
high, 'b+')
ax[1].set_ylabel('Fuzzy membership')
ax[1].set_title('Humidity')
ax[1].set_ylim(-0.05, 1.05)
ax[1].set_xlim(0, 10)

lgd2 = ax[1].legend([t1, t2], ['Hum. low', 'Hum. high'], loc='center
left', bbox to anchor=(1, 0.5))

plt.grid(True)
plt.tight_layout()
plt.show()
fig.savefig('fuzzy_mem_temp_hum.png', dpi=100, bbox_extra_artists=
(lgd1, lgd2, ), bbox_inches='tight')
print('done')
```

现在，准备好通过 DHT22 模块来读取温度和湿度数据。接着，把它们计算入模糊逻辑系统。

另外，从输入数据中做出模糊推断。通过模糊聚合来生成最终的输出。

输出是以数字形式。可以将它映射成低、平均还有十分舒适。从这种情况来看，可以做出一个关于是否想要打开制冷装置的决策。

```
# 检测和做出决策
print('program is ready for making decision based fuzzy logic')
machine_state = -1
try:
    while 1:
        print('sensing...')
        sen_humidity, sen_temperature = Adafruit_DHT.read_retry(sensor,
dht_pin)

        if humidity is not None and temperature is not None:
            print('Sensing: Temperature={0:0.1f}*C  Humidity={1:0.1f}%'.
format(sen_temperature, sen_humidity))

            sen_temperature = 18
            sen_humidity = 80
            # 归一化
            norm_temperature = sen_temperature / 60.0
            norm_humidity = sen_humidity / 100.0
            print('Normalization: Temperature={0:0.0001f}  Humidity=
{1:0.0001f}'
                  .format(norm_temperature, norm_humidity))

            temp_in = temperature_category(norm_temperature)
            hum_in = humidity_category(norm_humidity)
            print('fuzzy membership: Temperature={0}  Humidity={1}'.
format(temp_in, hum_in))

            # 决定权重并且聚合
            rule1 = np.fmax(temp_in['hot'], hum_in['low'])
            rule2 = temp_in['warm']
            rule3 = np.fmax(temp_in['warm'], hum_in['high'])

            imp1 = np.fmin(rule1, comfort_low)
            imp2 = np.fmin(rule2, comfort_ave)
            imp3 = np.fmin(rule3, comfort_very)
```

```
            aggregate_membership = np.fmax(imp1, imp2, imp3)

            # 去模糊化
            result_comfort = fuzz.defuzz(comfort, aggregate_membership,
'centroid')
            print(result_comfort)

            # 根据实验做出决策
            if result_comfort >= 5.002:
                if machine state < 0:
                    machine state = 1
                    print("turn on a machine")
                    GPIO.output(relay pin, GPIO.HIGH)
                else:
                    print("a machine already turn on")
            else:
                if machine_state > 0:
                    machine_state = 0
                    print("turn off a machine")
                    GPIO.output(relay_pin, GPIO.LOW)
                else:
                    print("a machine already turn off")

            time.sleep(2)

        time.sleep(2)

except KeyboardInterrupt:
    GPIO.output(relay_pin, GPIO.LOW)
    GPIO.cleanup()


print('program is exit')
```

把这些图片都保存。

测试

在写完一段程序之后，可以开始执行程序。输入下列命令：

```
$ sudo python ch02_fuzzy.py
```

确保 DHT22 和继电器模块都已经被添加到 Raspberry Pi。这段程序的一个示例输出如图 2-15 所示。

```
Documents — pi@raspberrypi: ~/Documents/book — ssh pi@192.168.0.12 — 8...
pi@raspberrypi:~/Documents/book $ sudo python ch02_fuzzy.py
initialization...
saving membership...
done
program is ready for making decision based fuzzy logic
sensing...
Sensing: Temperature=28.6*C  Humidity=85.0%
Normalization: Temperature=0.3  Humidity=0.8
fuzzy membership: Temperature={'hot': 6.8987413995925987e-19, 'warm': 5.44745042
4466365e-05, 'cold': 0.96078943915232318}  Humidity={'high': 4.6005175273896544e
-17, 'low': 1.0}
5.00202884593
turn on a machine
sensing...
Sensing: Temperature=28.6*C  Humidity=85.1%
Normalization: Temperature=0.3  Humidity=0.8
fuzzy membership: Temperature={'hot': 6.8987413995925987e-19, 'warm': 5.44745042
4466365e-05, 'cold': 0.96078943915232318}  Humidity={'high': 4.6005175273896544e
-17, 'low': 1.0}
5.00202884593
a machine already turn on
sensing...
Sensing: Temperature=28.6*C  Humidity=85.0%
Normalization: Temperature=0.3  Humidity=0.8
fuzzy membership: Temperature={'hot': 6.8987413995925987e-19, 'warm': 5.44745042
4466365e-05, 'cold': 0.96078943915232318}  Humidity={'high': 4.6005175273896544e
-17, 'low': 1.0}
5.00202884593
a machine already turn on
sensing...
Sensing: Temperature=28.6*C  Humidity=85.1%
Normalization: Temperature=0.3  Humidity=0.8
fuzzy membership: Temperature={'hot': 6.8987413995925987e-19, 'warm': 5.44745042
```

图 2-15

提高

该程序是展示如何使用模糊逻辑来设计一个决策系统的样例，可以尝试很多不同的提高该程序的方法。下面列出的是一些可以考虑的提高该程序的地方。

- 改变模糊 membership 模型来提高舒适度的定义。
- 添加更多输入数据来提高精确度。
- 添加模糊接口函数来获得聚合的数值。

总　结

本章已经通过两个样例：贝叶斯模型和模糊逻辑来描述了一些基本的决策系统，也探索了用来实现贝叶斯模型和模糊逻辑的 Python 函数库，并且亲手进行了实践。

最后，用模糊逻辑部署了一个决策系统。利用该学习样例，学习了如何把决策系统和 Raspberry Pi 的物联网项目相结合。

在第 3 章中，将要学习如何为物联网项目搭建一个视觉机器。

引　用

下面是一个推荐书籍的列表。在这些书籍中可以学习到更多关于本章的主题内容。

1. Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press. 2004.

2. Peter D. Hoff. *A First Course in Bayesian Statistical Methods*. Springer, New York. 2009.

3. James V Stone. *Bayes' Rule: A Tutorial Introduction to Bayesian Analysis*. Sebtel Press. 2013.

4. Matt Sekerke. *Bayesian risk management: a guide to model risk and sequential learning in financial markets*. Wiley & Sons. 2015.

5. Timothy J. Ross. *Fuzzy logic with engineering applications, 3rd Edition*. John Wiley & Sons. 2010.

6. Hung T. Nguyen and Elbert A. Walker. *A First Course in Fuzzy Logic, 3rd Edition*. CRC Press. 2006.

第 3 章　搭建机器视觉

眼睛是人们看到美丽世界的重要窗口。本章将探索如何通过部署一台摄像机来让机器能够看见东西。我们将会通过训练机器来检测和追踪一种物体，这样可以开始了解机器视觉的工作原理，同时也会回顾几个摄像机模块部件。

具体而言，会探索如下几个主题：

- 机器视觉的基本介绍
- 介绍 OpenCV 函数库
- 将 OpenCV 函数库部署到 Raspberry Pi 上
- 通过 OpenCV 设计一个简单的程序
- 学习如何使用摄像机模块
- 介绍用于机器视觉的模式识别
- 为移动的物体搭建一个最终视觉系统
- 搭建物联网机器视觉

机器视觉的基本介绍

视觉机器是具备摄像和理解所看见物体是什么的能力的一台机器。机器使用它的摄像机去检测它周围环境中的物理部件。机器视觉或者计算机视觉是一个具体领域。在该领域中，机器可以获取、分析并且理解一个静态的图像或者视频。该领域涉及诸如图像处理、模式识别和机器学习这样的知识。

模式识别和机器学习领域帮助人们训练机器去理解图像。例如，当展示一幅有人在一部汽车里的图像给一台机器，机器应该可以识别出哪一部分是人，哪一部分是汽车。另外，在某些情境下，机器应该也可以猜测图像中的人。从模式识别和机器学习的观点来看，应该用一个数据库注册一群特定的人，这样机器就能够在一张给定的图片中检测出有人之后判断这个人是谁。

为搭建机器视觉，使用如图 3-1 所示的大概设计流程。

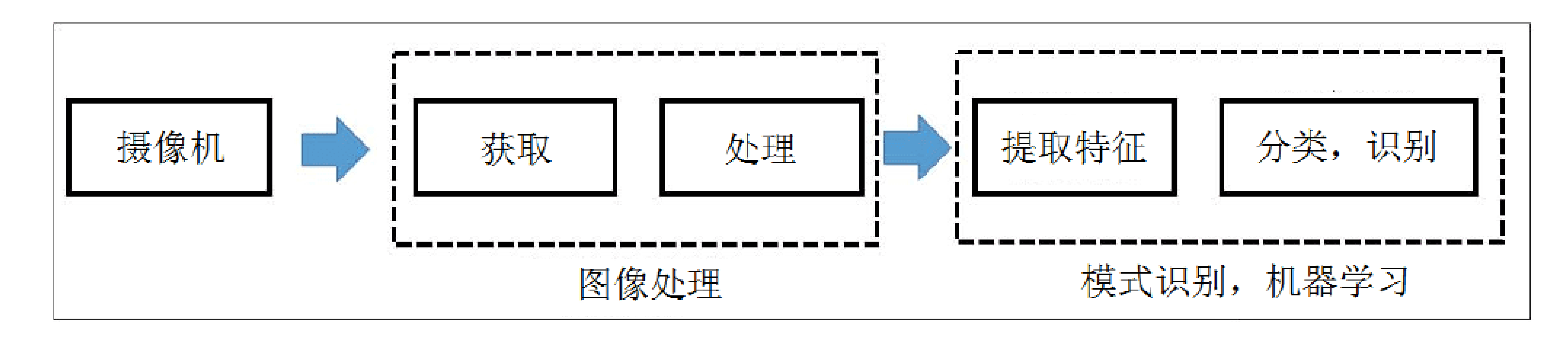


图 3-1

首先，从一台摄像机获取图像集合。每个图像将被诸如去燥、过滤或者转换这样的图像处理环节所处理。然后，为每幅图片做特征提取。

根据目的的不同，有很多不同的特征提取技术。在获取了图像的特征之后，检测并且识别出一幅图上的物体。模式识别和机器学习技术将会参与这个处理过程。

本书不会对模式识别和机器学习解释太多。推荐阅读关于模式识别和机器学习的教科书。在本章中，会展示如何通过在物联网设备上运用模式识别和机器学习来最终获得机器视觉。

OpenCV 函数库介绍

OpenCV（Open Computer Vision）函数库是一个为计算效率而设计的开源函数库，主要关注点在于及时类应用。该函数库由 C/C++编写并且为其他编程语言提供了几种捆绑接口。OpenCV 官方网站可通过网址 http://www.opencv.org 访问。

OpenCV 函数库提供了一套包括从基本计算和图像处理到模式识别和机器学习的完整的函数库。不少学术论文都使用该函数库进行模拟和实验，所以该函数库是一个非常好的出发点，可以用它来开始在机器视觉和计算机视觉上的项目。

目前，OpenCV 函数库可以在 Windows、Linux、Mac、Android 以及 iOS 平台上使用。可以通过网址 http://opencv.org/downloads.html 下载到该函数库。下面将会展示如何在带有 Raspbian OS 的 Raspberry Pi 上面配置 OpenCV 函数库。

在 Raspberry Pi 上配置 OpenCV

在本节中，将会在 Raspberry Pi 上配置 OpenCV 函数库。此处将会使用 Raspbian Jessie 来测试。下面将要在 Raspberry Pi 开发板上从源码开始安装 OpenCV 函数库。

首先，安装开发函数库。在 Raspberry Pi 终端中输入下面的命令：

```
$ sudo apt-get update
$ sudo apt-get install build-essential git cmake pkg-config libgtk2.0-dev
$ sudo apt-get install python2.7-dev python3-dev
```

同时也需要安装所需要的矩阵、图片还有视频函数库。可以输入下面的命令来进行安装：

```
$ sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev
libv4l-dev
$ sudo apt-get install libxvidcore-dev libx264-dev
$ sudo apt-get install libatlas-base-dev gfortran
```

下一步就是通过 Git 下载 OpenCV 源代码。可以输入下面的一系列命令：

```
$ mkdir opencv
$ cd opencv
$ git clone https://github.com/Itseez/opencv.git
$ git clone https://github.com/Itseez/opencv_contrib.git
```

这里使用一个 Python 虚拟环境 virtualenv 在 Raspberry Pi 上部署 OpenCV。这种方法的便利性在于它可以将已有的 Python 开发环境分离开来。

如果 Raspbian 中还没有安装这个虚拟环境，可以通过 pip 来安装。

```
$ sudo pip install virtualenv virtualenvwrapper
$ sudo rm -rf ~/.cache/pip
```

安装完成之后，可以在 bash profile 中对 virtualenv 进行配置：

```
$ nano ~/.profile
```

然后，将下面的脚本添加进 profile 文件：

```
export WORKON HOME=$HOME/.virtualenvs
source /usr/local/bin/virtualenvwrapper.sh
```

完成之后，将 bash profile 文件进行保存。

要创建一个 Python 虚拟环境，可以输入下面的命令：

```
$ mkvirtualenv cv
```

该命令将会创建一个叫作 cv 的 Python 虚拟环境。

如果使用 Python 3，可以通过下面的命令来创建该虚拟环境：

```
$ mkvirtualenv cv -p python3
```

应该可以在终端上看到（cv）。如果关闭了该终端或者开启了一个新的终端，应该重新激活 Python 虚拟环境。输入下面这些命令：

```
$ source ~/.profile
$ workon cv
```

在图 3-2 中，可以看到一个叫作 cv 的 Python 虚拟空间的示例。

```
agusk — pi@raspberrypi: ~ — ssh pi@192.168.0.12 — 80×21
pi@raspberrypi:~ $ source ~/.profile
pi@raspberrypi:~ $ workon cv
(cv) pi@raspberrypi:~ $
```

图 3-2

在 Python 虚拟终端内部，继续安装 NumPy 作为 OpenCV 所需要的函数库。可以使用 pip 命令安装该函数库：

```
$ pip install numpy
```

现在已经准备好来从源代码开始构建并且安装 OpenCV 函数库。在复制了 OpenCV 函数库之后，可以通过输入下面这些命令来构建它：

```
$ cd ~/opencv/
$ mkdir build
$ cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \
  -D CMAKE_INSTALL_PREFIX=/usr/local \
  -D INSTALL_C_EXAMPLES=ON \
  -D INSTALL_PYTHON_EXAMPLES=ON \
  -D OPENCV_EXTRA_MODULES_PATH=~/opencv/opencv_contrib/modules \
  -D BUILD_EXAMPLES=ON ..
```

另外，在 Raspbian OS 的内部系统中安装 OpenCV 函数库。

```
$ make -j4
$ sudo make install
$ sudo ldconfig
```

如果这些都完成以后，应该配置该函数库，这样 Python 就可以通过 Python binding

来访问到这些函数库。下面显示的一系列命令是在 Python 2.7 上进行配置的具体步骤：

```
$ ls -l /usr/local/lib/python2.7/site-packages/
$ cd ~/.virtualenvs/cv/lib/python2.7/site-packages/
$ ln -s /usr/local/lib/python2.7/site-packages/cv2.so cv2.so
```

如果使用 Python 3.x，如 Python 3.4，那么可以在终端中进行下面的步骤。这里考虑使用的是 Python 3.4.x：

```
$ ls /usr/local/lib/python3.4/site-packages/
$ cd /usr/local/lib/python3.4/site-packages/
$ sudo mv cv2.cpython-34m.so cv2.so
$ cd ~/.virtualenvs/cv/lib/python3.4/site-packages/
$ ln -s /usr/local/lib/python3.4/site-packages/cv2.so cv2.so
```

安装过程已经结束。现在需要通过检查 OpenCV 版本来验证是否正确地安装了 OpenCV：

```
$ workon cv
$ python
>>> import cv2
>>> cv2.__version__
```

应该在终端中看到 OpenCV 版本号。图 3-3 显示了程序输出的一个示例。

```
agusk — pi@raspberrypi: ~/opencv — ssh pi@192.168.0.12 — 80×21
(cv) pi@raspberrypi:~/opencv $ python
Python 2.7.9 (default, Mar  8 2015, 00:52:26)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'3.1.0-dev'
>>>
```

图 3-3

接下来的示例使用 OpenCV 展示了一个图片文件。对于这个应用场景，可以使用 cv2.imshow()函数来显示一个图片文件。

对于测试，可以登录 Raspberry Pi 桌面来执行该程序。用一个文本编辑器，可以输入下面的脚本：

```
import numpy as np
import cv2
```

```
img = cv2.imread('circle.png')
cv2.imshow('My photo', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

此处使用 circle.png 文件作为图片源。可以在本书的源代码里找到这幅图片。将这些脚本保存到一个文件并命名为 ch03_hello_opencv.py。接着，打开 Raspberry Pi 桌面的终端并且输入下面的命令：

```
$ python ch03_hello_opencv.py
```

如果命令执行成功，应该可以看到一个显示了一幅图片的对话框，如图 3-4 所示。

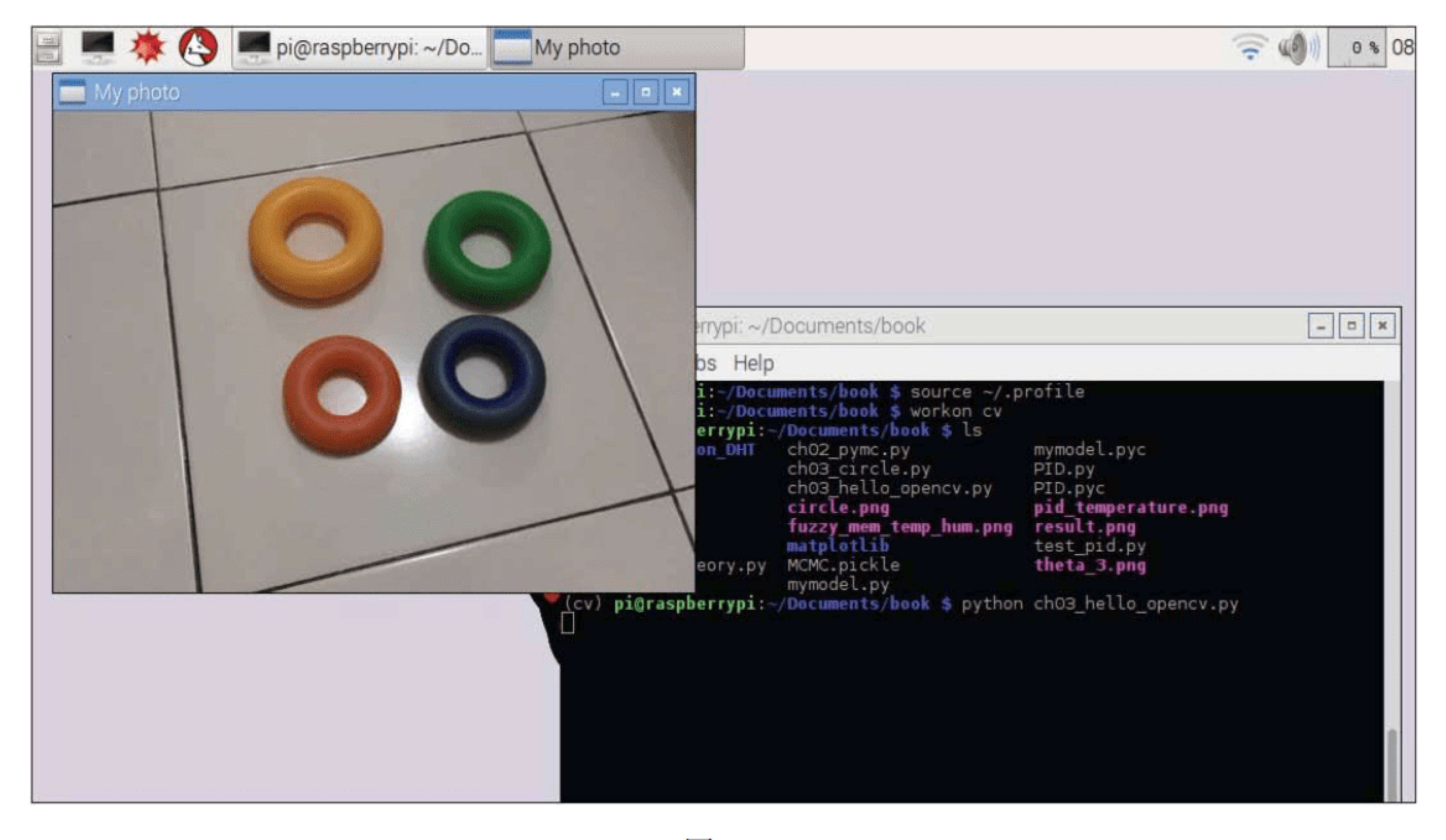


图 3-4

因为在代码中调用了 cv2.waitKey(0)，所以这个图片对话框会出现。现在如果想要关闭这个对话框，可以在图片对话框里面单击任何按钮。

在收到一个鼠标单击事件之后，通过调用 cv2.destroyAllWindows()函数关闭该图片对话框。

使用 OpenCV 编写一个简单的程序

有很多程序示例可以教会用户如何在 Python 上使用 OpenCV。在我们的用例中，编写一个简单的程序来检测一幅静态图像中的圆圈。

考虑现在有如图 3-5 所示的一幅图片用于测试。可以在源代码文件里面找到这个名为 circle.png 的图片。



图 3-5

为了找到一幅静态图片中的圆圈，使用 Circle Hough Transform（CHT）。一个圆圈可以按如下方式进行定义：

$$(x-a)^2+(y-b)^2=r^2$$

(a,b)是该圆圈的中心，圆圈的半径为 r。这些参数将会通过 CHT 方法来被计算。

让我们开始搭建一个示例程序！

我们将编写一个程序来读入一个图片文件。接着，将要使用 cv2.HoughCircles()函数来检测一幅静态图片中的圆圈。

下面开始写下如下所示的脚本：

```
import cv2
import numpy as np
```

```
print('load image')
orig = cv2.imread('circle.png')
processed = cv2.imread('circle.png', 0)
processed = cv2.medianBlur(processed, 19)

print('processing...')
circles = cv2.HoughCircles(processed, cv2.HOUGH_GRADIENT, 1, 70,
                           param1=30,
                           param2=15,
                           minRadius=0,
                           maxRadius=50)


circles = np.uint16(np.around(circles))
for (x, y, r) in circles[0, :]:
    cv2.circle(orig, (x, y), r, (0, 255, 0), 2)

print('completed')
print('writing to a file..')
cv2.imwrite('circle_process.png', orig)
print('done')
```

将这些脚本保存到一个文件，并且命名为 ch03_circle.py。

为了运行这个程序，在 Raspberry Pi 终端里面输入下面的命令：

```
$ python ch03_circle.py
```

请务必确保文件 circle.png 和文件 ch03_circle.py 被放置在同一个目录下。

在终端里面应该会看到一些文字。可以在图 3-6 中看到该程序输出结果的一个示例。

```
agusk — pi@raspberrypi: ~/Documents/book — ssh pi@192.168.0.12 — 80×21
(cv) pi@raspberrypi:~/Documents/book $ ls
Adafruit_Python_DHT    ch02_pymc.py          mymodel.pyc
alpha.png              ch03_circle.py        PID.py
beta.png               ch03_hello_opencv.py  PID.pyc
ch01_dht22.py          circle.png            pid_temperature.png
ch01_led.py            fuzzy_mem_temp_hum.png result.png
ch01_pid.py            matplotlib            test_pid.py
ch02_bayes_theory.py   MCMC.pickle           theta_3.png
ch02_fuzzy.py          mymodel.py
(cv) pi@raspberrypi:~/Documents/book $ python ch03_circle.py
load image
processing...
completed
writing to a file..
done
(cv) pi@raspberrypi:~/Documents/book $
```

图 3-6

这个程序将会在一幅图片中检测是否有圆圈形式的存在。检测过程结束之后，该程序会生成一幅名为 circle_process.png 的新图片。

如果打开 circle_process.png 文件，应该可以看到图片文件中有 4 个圆圈被勾勒出来，如图 3-7 所示。



图 3-7

这些代码是如何工作并且生效的？

首先，把 OpenCV 和 NumPy 函数库加载到程序：

```
import cv2
import numpy as np
```

通过 cv2.imread()把图片读取到 orig 和 processed 两个变量中。processed 变量是用来找到圆圈的。由于模糊化的过程，processed 变量中的图片将会有所变化。

```
orig = cv2.imread('circle.png')
processed = cv2.imread('circle.png', 0)
processed = cv2.medianBlur(processed, 19)
```

cv2.medianBlue()是用来通过定义一个中位数来模糊化一幅图片的。参数数值应该是奇数，如 1, 3, 5, 7。

为了能在一幅图片中找到圆圈，可以使用 cv2.HoughCircles().param1 和 param2 数值，这两个数值根据论文 http://www.bmva.org/bmvc/1989/avc-89-029.pdf 所定义。

```
circles = cv2.HoughCircles(processed, cv2.HOUGH_GRADIENT, 1, 70,
                           param1=30,
                           param2=15,
                           minRadius=0,
                           maxRadius=50)
```

画出原始图片中所找到的所有圆圈，也就是 orig 变量：

```
circles = np.uint16(np.around(circles))
for (x, y, r) in circles[0, :]:
    cv2.circle(orig, (x, y), r, (0, 255, 0), 2)
```

最后一个步骤就是使用 cv2.imwrite()把计算结果保存到一个文件并命名为 circle_process.png：

```
cv2.imwrite('circle_process.png', orig)
```

使用摄像机模块

在本节中，会探索 Raspberry Pi 开发板的各种摄像机模块。有很多适合项目的摄像机模块。摄像机模块可以被看作是基于某种 Raspberry Pi 接口并且被附加的一个模块。

让我们开始探索吧。

基于 CSI 接口的摄像机模块

Raspberry Pi Camera 是被 Raspberry Pi 基金会发布的官方正式的摄像机模块。这个摄像机可以通过 CSI 接口被附加到 Raspberry Pi 开发板上。Raspberry Pi 基金会同时也通过了另外一个摄像机模块：Raspberry Pi NoIR Camera。该模块可以在低亮度（twilight）环境中工作。Raspberry Pi Camera v2 和 NoIR camera v2 的一种形式如图 3-8 所示。

这些模块是 Raspberry Pi 的官方摄像机模块。为了能通过 CSI 接口使用摄像机模块，应该在 Raspbian 上启用它。可以通过使用 raspi-config 工具来进行配置。只需要在 Raspberry Pi 命令行中输入如下命令即可：

```
$ sudo raspi-config
```

在执行之后，应该可以看到 raspi-config 程序，如图 3-9 所示。

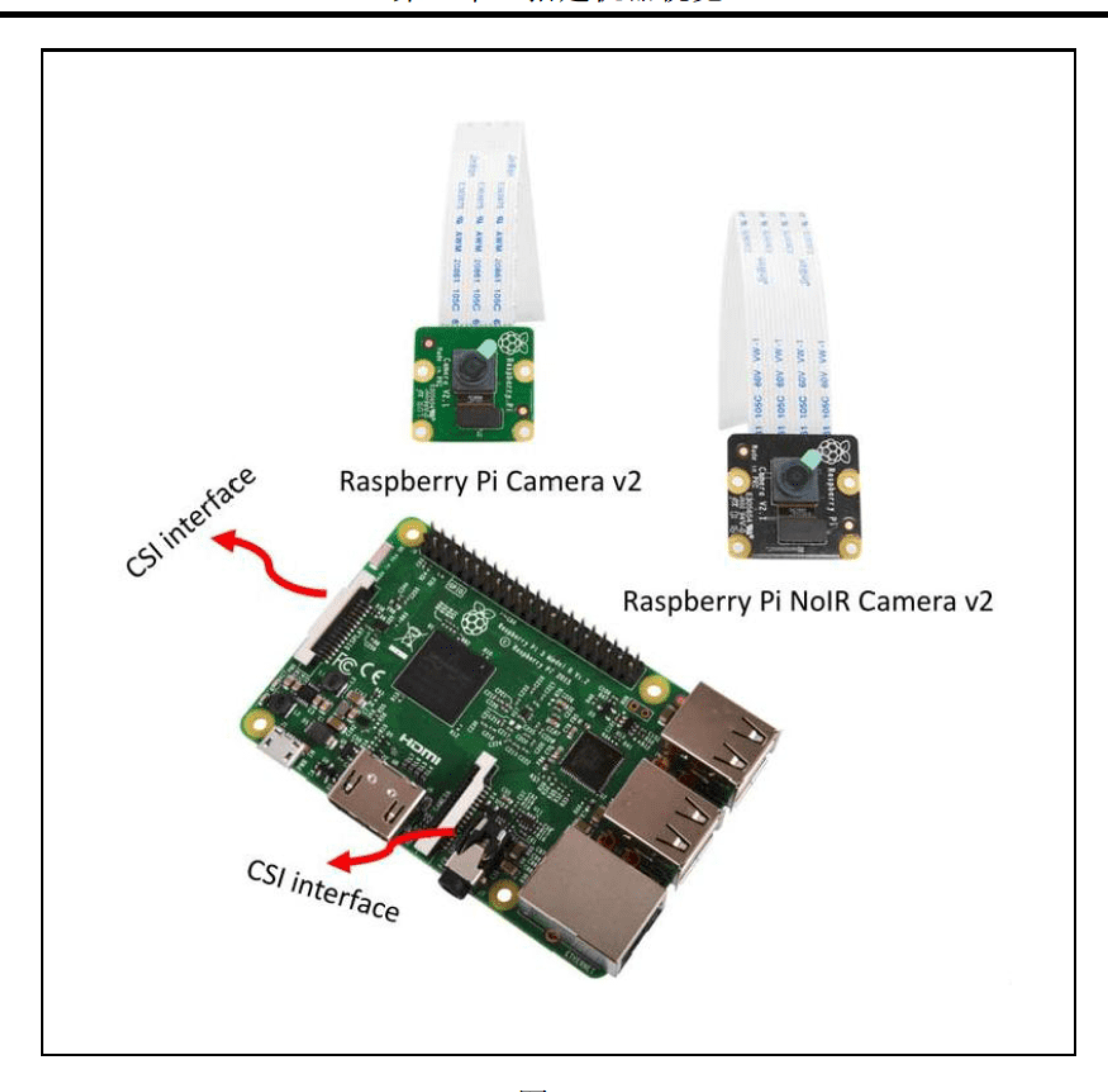


图 3-8

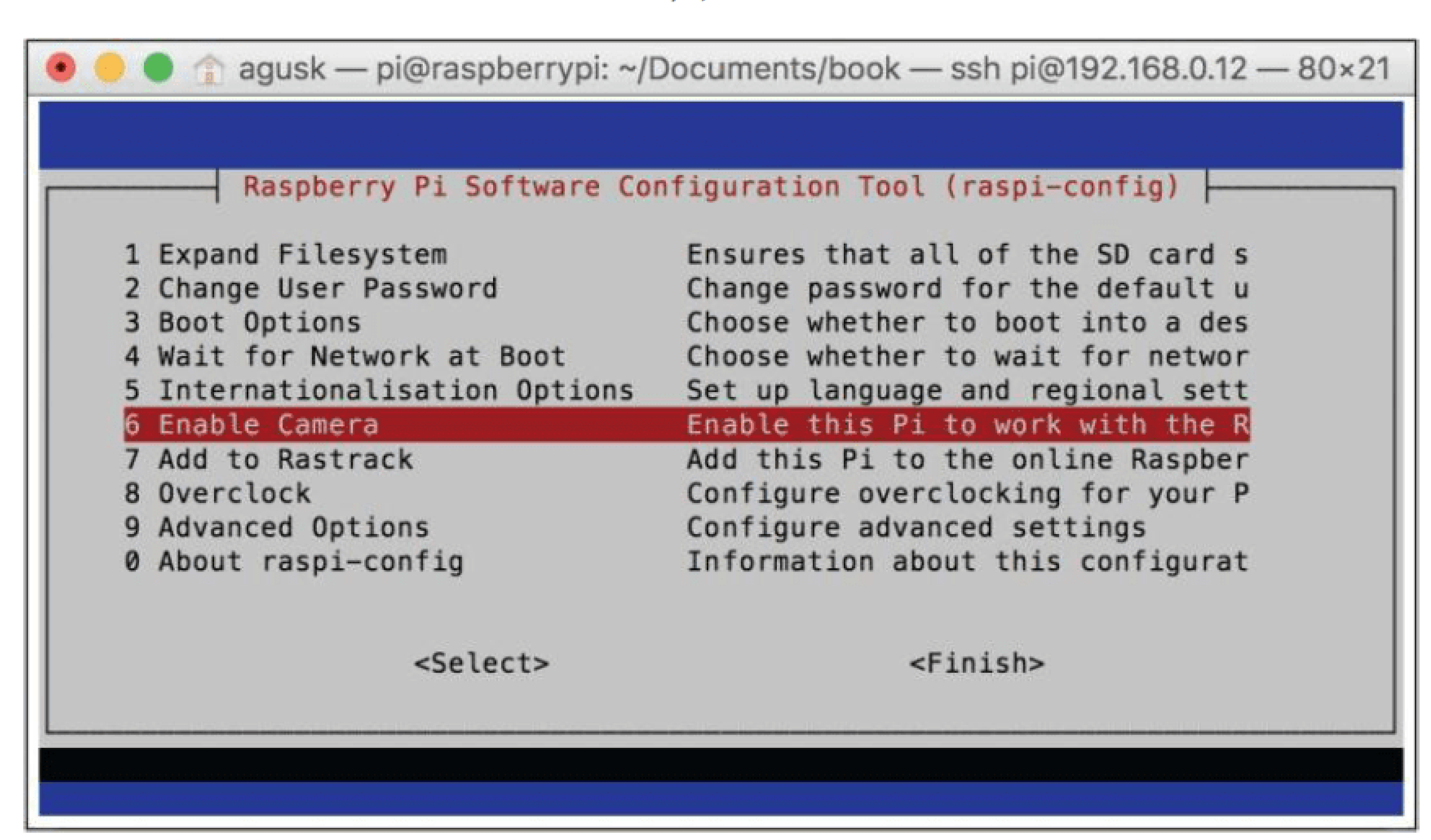


图 3-9

在 raspi-config 工具上选择 6 Enable Camera 选项，然后单击 Enable Camera 来激活摄

像机模块。完成之后，应该会被询问重新启动 Raspbian。重新启动 Raspbian 来完成配置的改变。

现在可以通过程序来使用摄像机模块了。

基于 USB 接口的摄像机模块

基于 USB 接口的摄像机模块是一种很普遍的摄像机设备，如图 3-10 所示。这种设备通常被叫作 webcam，可以轻易地在本地存储中找到它们。



来源：http://www.amazon.com/Microsoft-LifeCam-Cinema-Webcam-Business/dp/B004ABQAFO/

图 3-10

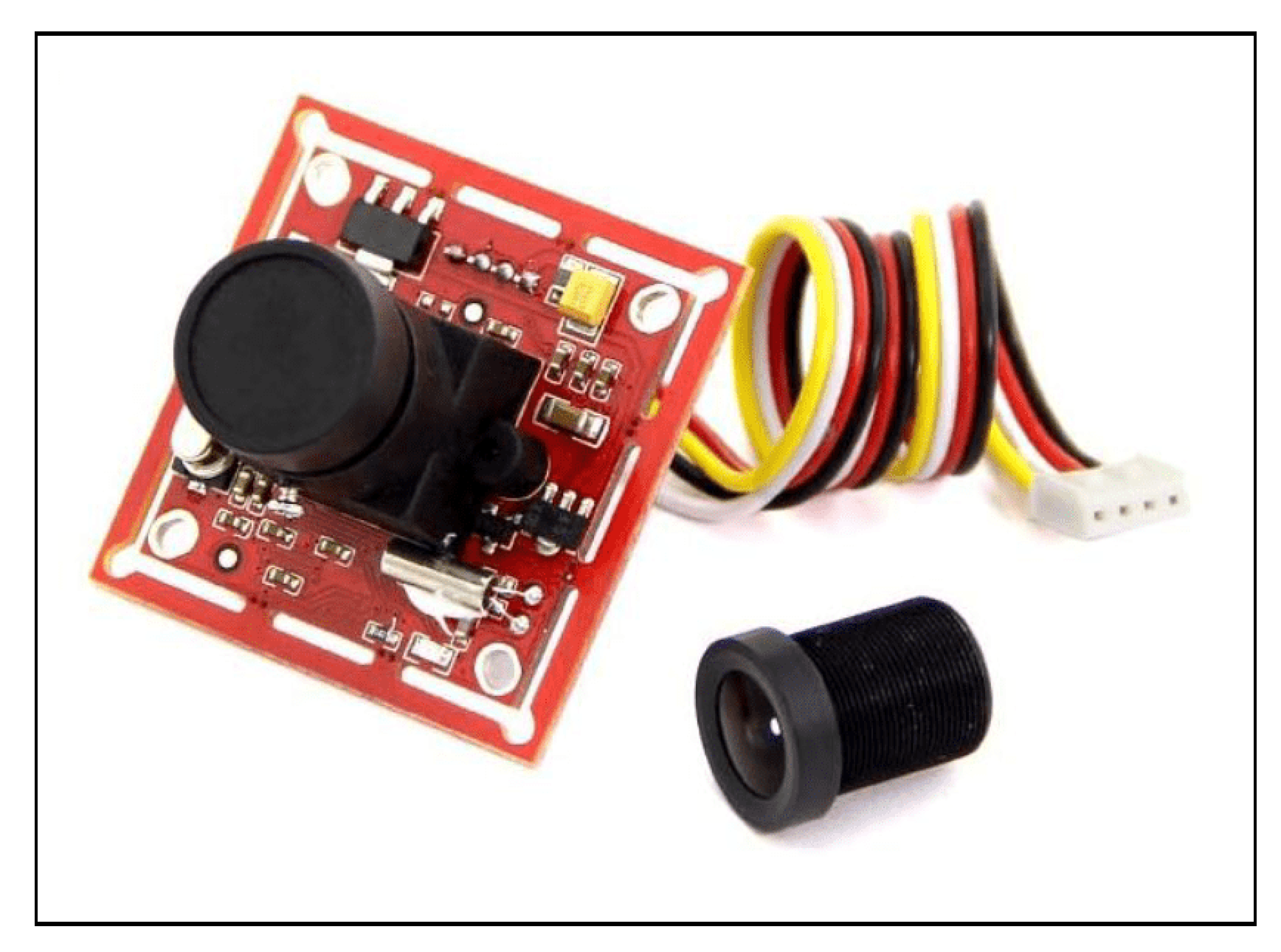
一个基于 USB 的摄像机模块可以通过 USB 被附加到 Raspberry Pi 开发板上。对于 Raspberry Pi 3，有 4 个 USB 适配器，所以可以基于 USB 添加 4 个摄像机模块。

几种基于 USB 的摄像机模块可以被包含了 OpenCV 函数库在内的 Raspberry Pi 开发板所识别。可以在网址 http://elinux.org/RPi_USB_Webcams 中找到兼容 Raspberry Pi 开发板的 USB webcams。

基于串行（serial）接口的摄像机模块

如果物联网开发板没有一个 USB 接口，但是有 UART/serial pins，可以使用一个基于

串行接口的摄像机模块。Grove-Serial Camera 组件是其中之一，如图 3-11 所示。



来源：http://www.seeedstudio.com/item_detail.html?p_id=1608

图 3-11

基于 UART 接口的摄像机模块可以通过 UART GPIO pins 被添加到 Raspberry Pi 开发板。

多种接口的摄像机模块

笔者找到了几个支持多种接口的摄像机模块，它们同时支持 serial、USB、SPI 还有 I2C 接口，这点非常好，因为可以把它们添加到任意最喜欢的开发板上。

Pixy CMUcam 是一种可以支持多种接口的摄像机模块，可以在官方网站 http://cmucam.org 上阅读和购买该模块。一些在线商城也会贩售这个模块。笔者从 SeeedStudio（网址为 http://www.seeedstudio.com）上买到了 Pixy CMUcam5 开发板还有 pan/tilt 模块，如图 3-12 所示。

在最后一节中将会和读者分享如何在 Raspberry Pi 开发板上使用 Pixy CMUcam5 模块。



图 3-12

从 OpenCV 函数库访问摄像机模块

在使用摄像机模块这一章中，使用了一幅静态图片作为 OpenCV 函数库的输入源。除此之外，还可以用摄像机作为静态图像的输入源。摄像机生成视频数据，视频数据包含了一系列静态图片。为了从 OpenCV 函数库访问摄像机模块，请遵照如下步骤：

（1）为了从 OpenCV 访问摄像机模块，可以使用 VideoCapture 对象。调用 read()来读取一帧，这一帧便是一幅静态图像。

（2）为了演示，使用摄像机 U 盘。只需要通过 U 盘把该设备连接到 Raspberry Pi 开发板上，然后用文本编辑器写出下列脚本：

```
import numpy as np
import cv2


cap = cv2.VideoCapture(0)
while True:
    # 一帧一帧地进行捕捉
```

```
    ret, frame = cap.read()

    # 显示结果返回的帧
    cv2.imshow('video player', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break


cap.release()
cv2.destroyAllWindows()
```

（3）将这些脚本保存为一个文件并命名为 ch03_camera_player.py。

（4）为了运行该程序，应该进入 Python 虚拟环境，该虚拟环境应该已经部署了 OpenCV 函数库。

（5）输入下面这个命令：

```
$ python ch03_camera_player.py
```

（6）成功后可以看到一个对话框展示来自摄像头的视频流。图 3-13 所示是一个示例输出。

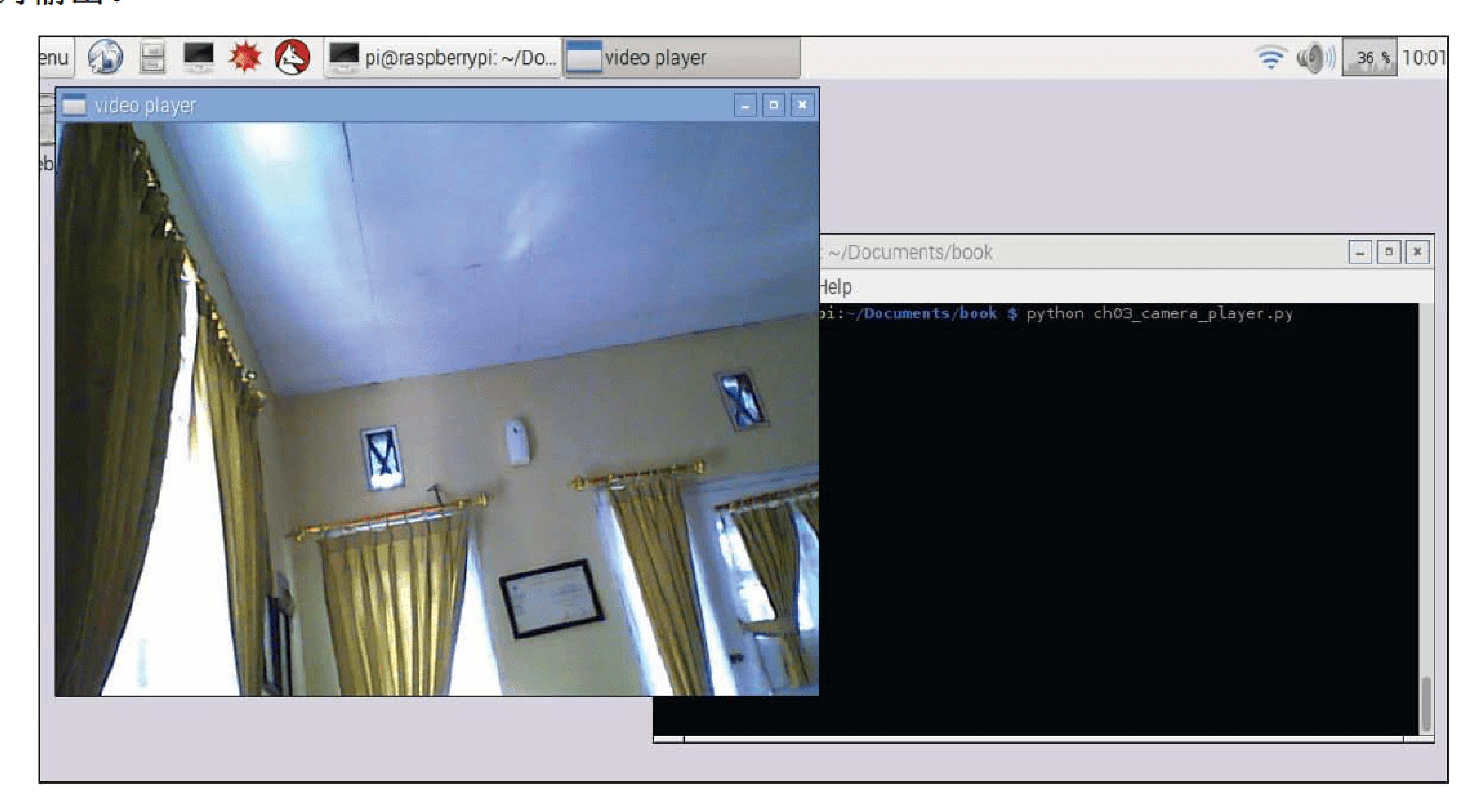


图 3-13

（7）按下字母 Q 退出或者关闭对话框。

（8）函数 cv2.VideoCapture(0)调用添加的摄像机器。如果添加了两个摄像机，用函数 cv2.VideoCapture(1)调用第二个。

介绍用于机器视觉的模式识别

模式识别是机器视觉或者计算机视觉的重要部分，用来告诉机器如何理解图像中的物体。

在本章中，将研究一篇由 Paul Viola 和 Michael Jones 写的论文：*Rapid Object Detection using a Boosted Cascade of Simple Features*。这篇论文描述了用于视觉物体检测的机器学习方法。

一般来说，Viola 和 Jones 的方法也被称作 Haar Cascades。他们使用带有如下分类器的 AdaBoost 算法：

$$h(x)=\begin{cases}1 & \sum_{t=1}^{T}\alpha_t h_t(x)\geqslant\frac{1}{2}\sum_{t=1}^{T}\alpha_t\\ 0 & otherwise\end{cases}$$

幸运的是，OpenCV 库已经实现了 Viola 和 Jones 的方法用来做视觉物体检测。其他人也对数据训练做出了贡献。可以在源代码里找到训练文件，位置在 <opencv_source_codes>/data/haarcascades/。

现在可以使用 Haar Cascades 方法在图像上测试人脸识别。可以编写如下脚本：

```
import numpy as np
import cv2


face cascade = cv2.CascadeClassifier ('haarcascade frontalface
default.xml')

img = cv2.imread('children.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = face_cascade.detectMultiScale(gray, 1.3, 5)
for (x, y, w, h) in faces:
    img = cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 255), 2)
```

```
cv2.imshow('img', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

保存代码为 ch03_faces.py。

也可以把文件 haarcascade_frontalface_default.xml 和 children.png 放在程序的同一路径下。haarcascade_frontalface_default.xml 可以在<opencv_source_codes>/data/haarcascades/路径下找到，children.png 文件在本书的源码文件中。

在 Raspberry Pi 终端上运行如下命令：

```
$ python ch03_faces.py
```

之后可以看到一个带有图片的对话框。有 3 张人脸被识别，但是遗漏了一个。在笔者看来，原因是 Haar Cascades 方法虽然好，但不是最佳的方法。图 3-14 所示是程序运行后的示例。



图 3-14

程序的原理是什么呢？

首先为 Haar Cascades 加载需要的库和训练数据：

```
import numpy as np
import cv2
face_cascade = cv2.CascadeClassifier ('haarcascade_frontalface_
default.xml')
```

载入图片用于测试并修改图片的颜色为灰色：

```
img = cv2.imread('children.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

为了识别人脸，调用 face_cascade.detectMultiScale()函数并传给函数图像的向量，比例因子和最小邻居数作为参数。如果识别出一张人脸，便在图片上画一个方框：

```
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
for (x, y, w, h) in faces:
    img = cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 255), 2)
```

最后是展示图片，然后等待用户按下任意键退出：

```
cv2.imshow('img', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

为移动的物体搭建视觉识别系统

在本节中，将搭建一个简单的视觉跟踪系统。我们已经学习了如何在图片中识别人脸，现在学习如何在视频中识别。

方法很简单。把输入的静态图变为摄像机传送来的一帧帧的图像。之后从 VideoCapture 对象中调用 read()方法，把一帧帧的图像传给函数 face_cascade.detectMultiScale()，然后在对话框中展示。

输入如下代码：

```
import numpy as np
import cv2

face_cascade = cv2.CascadeClassifier ('haarcascade_frontalface_
default.xml')
cap = cv2.VideoCapture(0)
while True:
    # 一帧帧捕捉
    ret, frame = cap.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

```
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    for (x, y, w, h) in faces:
        img = cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255,
255), 2)

    cv2.imshow('face tracking', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break


cap.release()
cv2.destroyAllWindows()
```

保存代码到文件 ch03_faces_camera.py。

在 Raspberry Pi 终端里执行命令：

```
$ python ch03_faces_camera.py
```

运行后，展示自己的脸，然后程序就可以识别出来，可以看到程序如图 3-15 所示的输出。

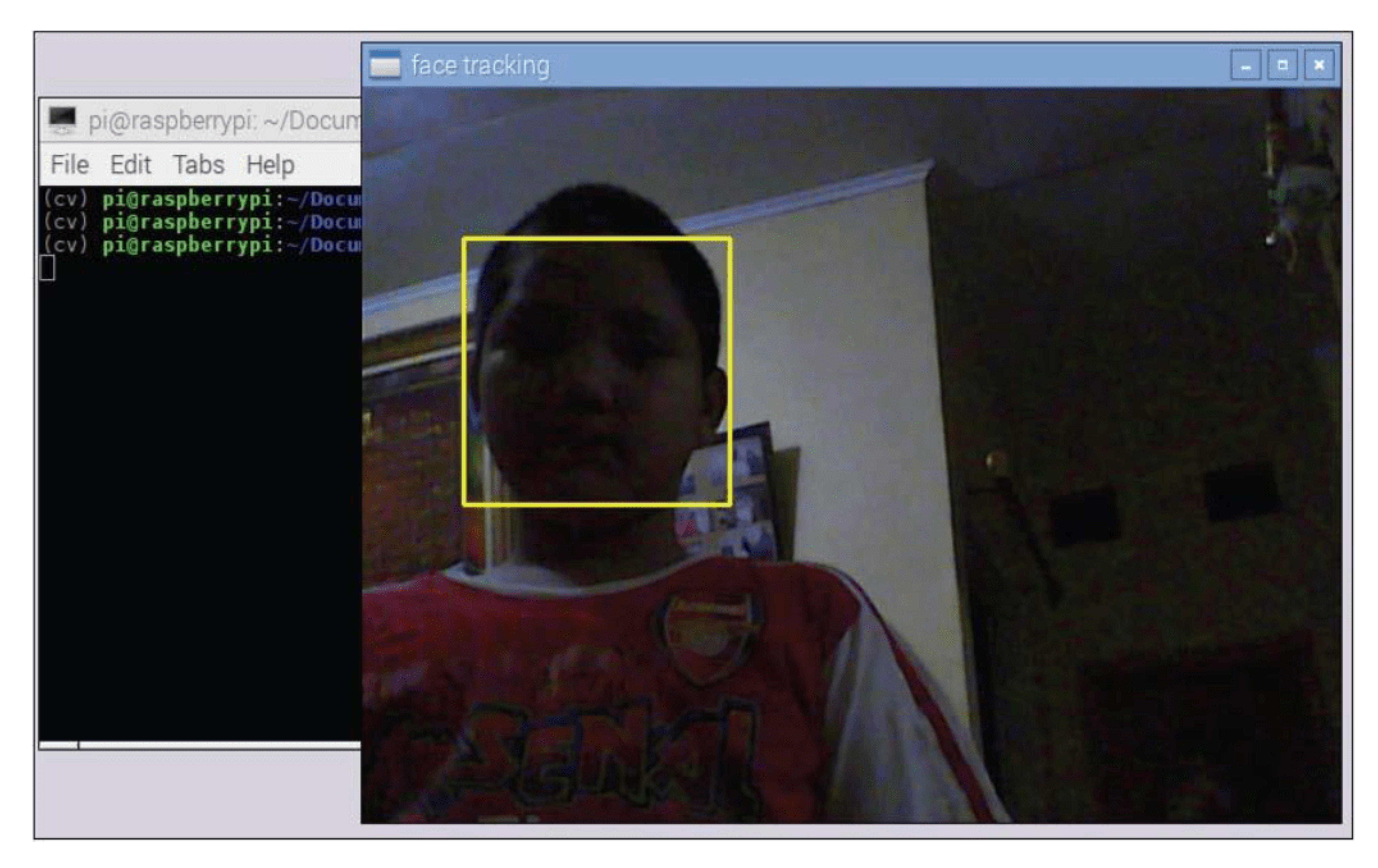


图 3-15

可以修改项目添加一个 LED 用来指示已经被识别的人脸。

如何实现的呢？

程序和之前的一样，只不过把输入变成了摄像机。

搭建 IoT 机器视觉

因前面的内容已经了解 Pixy CMUcam5 可以作为摄像机模块。本节将在 IoT 项目中使用它。

下面是一些需要的模块。

- Pixy CMUcam5 传感器：http://www.seeedstudio.com/item_detail.html?p_id=2048。
- Pan/Tilt for Pixy：http://www.seeedstudio.com/item_detail.html?p_id=2048。

也可以在其他在线商店购买。

在 Raspberry Pi 上部署 Pixy CMUcam5

为了使用 Pixy CMUcam5，应该安装需要的库和应用。首先打开 Raspberry Pi 的终端，输入如下命令：

```
$ sudo apt-get install libusb-1.0-0-dev
$ sudo apt-get install qt4-dev-tools
$ sudo apt-get install qt4-qmake qt4-default
$ sudo apt-get install g++
$ sudo apt-get install swig
$ sudo apt-get install libboost-all-dev
```

需要 Pixy 和应用的源码。首先下载源代码并安装 PixyMon：

```
$ git clone https://github.com/charmedlabs/pixy.git
$ cd pixy/scripts
$ ./build_pixymon_src.sh
```

为了用 Python 执行 Pixy 库，需要安装 Python binding。在路径<pixy_library>/pixy/scripts 下执行命令：

```
$./build_libpixyusb_swig.sh
```

需要配置用权限通过 USB 访问 Pixy。输入如下命令：

```
$ cd ../src/host/linux/
$ sudo cp pixy.rules /etc/udev/rules.d/
```

现在可以使用 Pixy CMUcam5。

装配

为了安装 Pixy CMUcam5 和 Pan/Tilt，推荐阅读安装指南 http://cmucam.org/projects/cmucam5/wiki/Assembling_pantilt_Mechanism。图 3-16 所示是安装后的样子。



图 3-16

升级 Pixy CMUcam5 固件

在使用 Pixy CMUcam5 模块之前，建议升级一下 IoT 板的固件。可以到 http://cmucam.org/projects/cmucam5/wiki/Latest_release 下载。例如，可以在 http://cmucam.org/attachments/download/1317/pixy_firmware-2.0.19-general.hex 直接下载 Pixy firmware 2.0.19。

为了升级固件，需要运行 PoxyMon 应用。先从 Raspberry Pi 上拔下 Pixy CMUcam5，然后按下 Pixy CMUcam5 顶部的白色按钮，再通过 USB 插回 Raspberry Pi。持续按住白色按钮直到出现一个文件夹对话框。选择 Pixy 固件文件，然后等待传输完成。

测试

开始用 Raspberry Pi 测试 Pixy CMUcam5。下面给出一些例子来展示如何使用 Pixy CMUcam5，让我们开始吧！

载入视频流

部署了 Pixy CMUcam5 应用和库之后，将获得 PixyMon 应用。这是用来管理训练数据的工具，可以在<pixy_codes>/build/pixymon/bin/路径下找到。

转到<pixy_codes>/build/pixymon/bin/，然后在 Raspberry Pi 终端输入：

```
$ ./PixyMon
```

如果执行成功，可以看到如图 3-17 所示的 PixyMon 窗口。

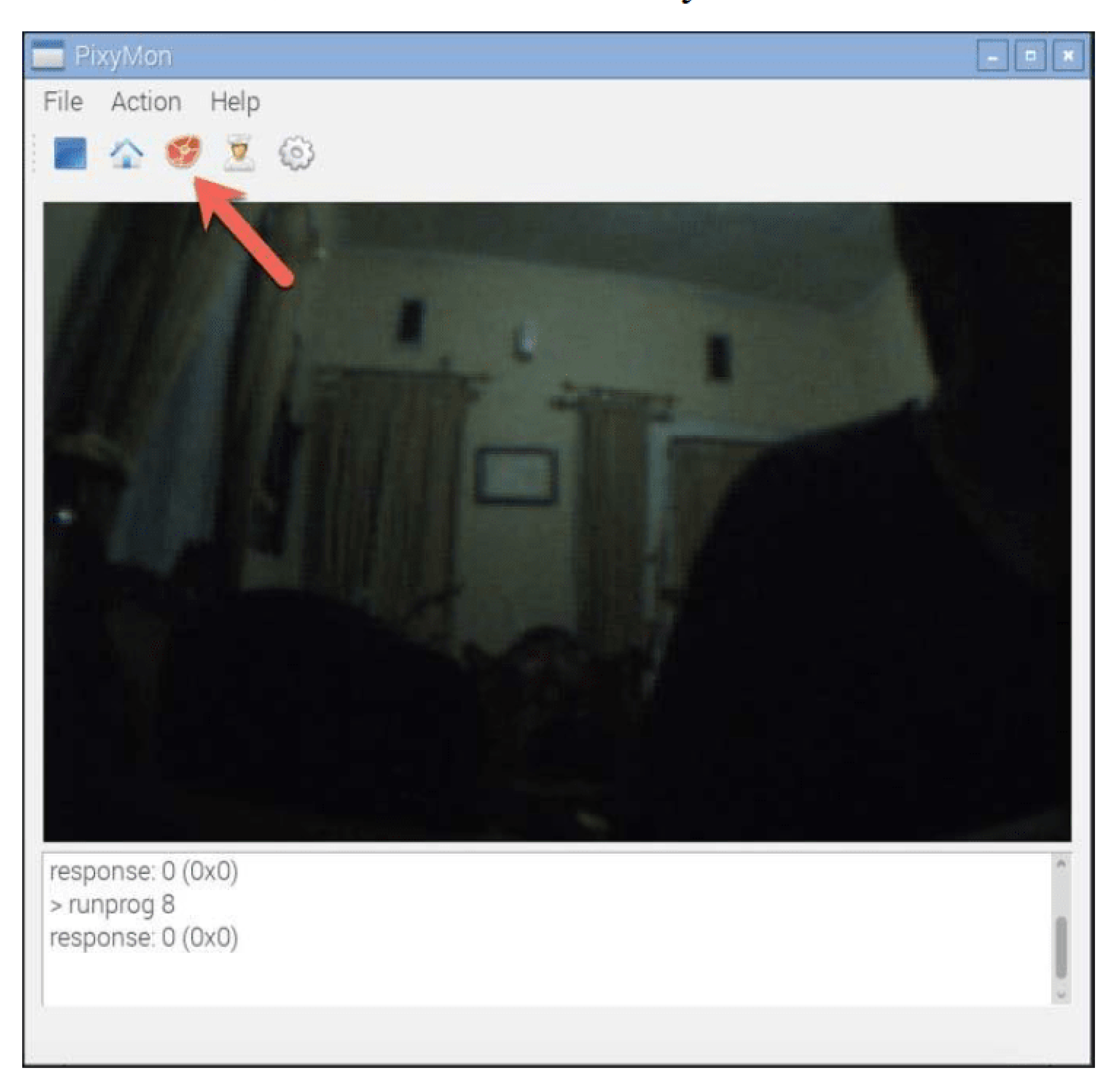


图 3-17

如果在窗口中没有看到任何图片，可以单击下面箭头指示的红色圆按钮，这会让 PixyMon 进入视频流模式。图 3-18 所示是程序的示例输出。

跟踪物体

Pixy CMUcam5 可以跟踪任意目标物体，只要这个物体被注册了。本节将探索如何注册一个新物体并跟踪它。

（1）将 Pixy CMUcam5 插入到 Raspberry Pi。打开 PixyMon 应用，展示任意想要追踪的物体，如图 3-19 所示。

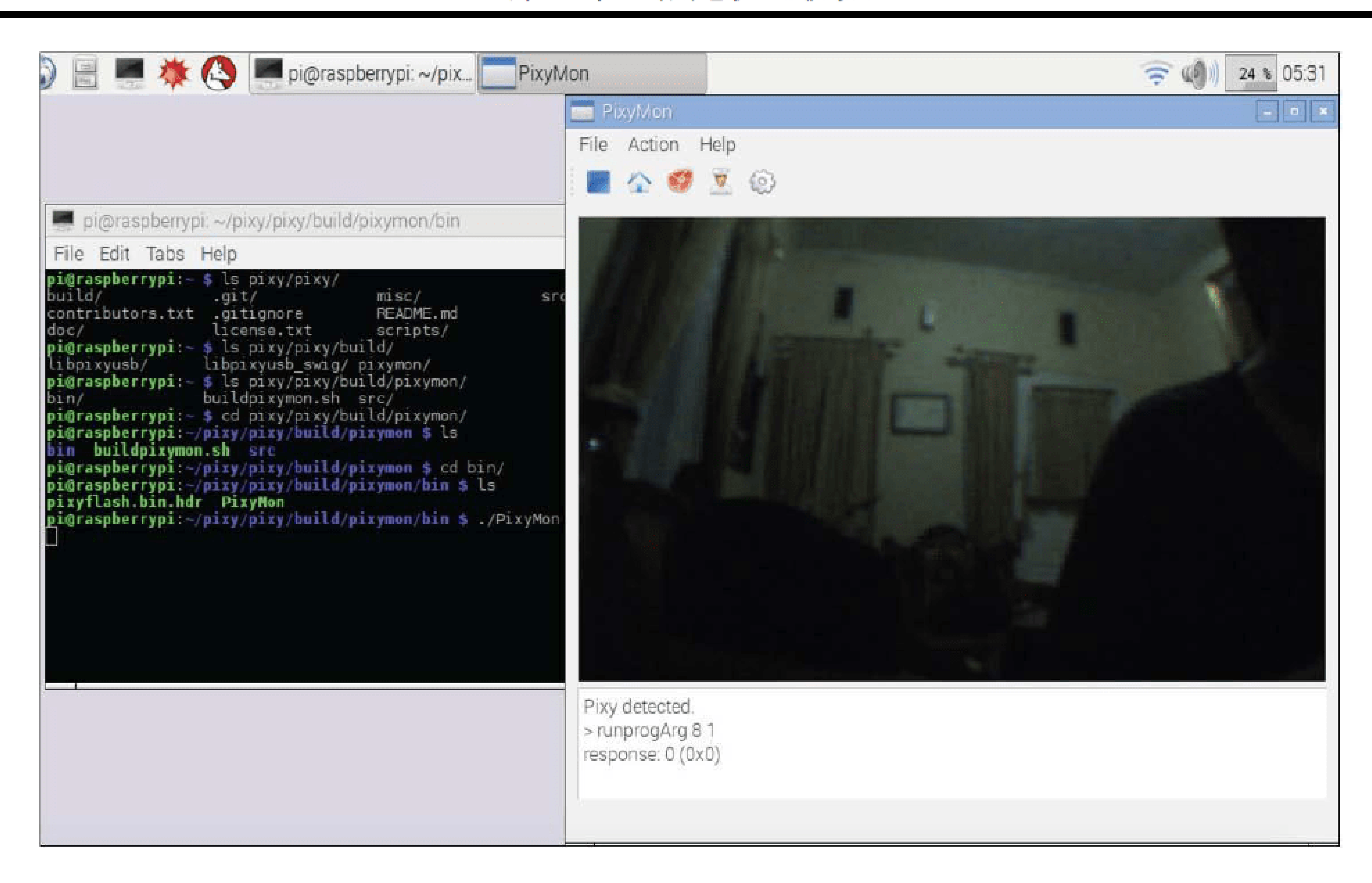

pi@raspberrypi: ~/pix...
PixyMon
24 %
05:31
pi@raspberrypi: ~/pixy/pixy/build/pixymon/bin
File Edit Tabs Help
pi@raspberrypi:~ $ ls pixy/pixy/
build/ .git/ misc/
contributors.txt .gitignore README.md
doc/ license.txt scripts/
pi@raspberrypi:~ $ ls pixy/pixy/build/
libpixyusb/ libpixyusb_swig/ pixymon/
pi@raspberrypi:~ $ ls pixy/pixy/build/pixymon/
bin/ buildpixymon.sh src/
pi@raspberrypi:~ $ cd pixy/pixy/build/pixymon/
pi@raspberrypi:~/pixy/pixy/build/pixymon $ ls
bin buildpixymon.sh src
pi@raspberrypi:~/pixy/pixy/build/pixymon $ cd bin/
pi@raspberrypi:~/pixy/pixy/build/pixymon/bin $ ls
pixyflash.bin.hdr PixyMon
pi@raspberrypi:~/pixy/pixy/build/pixymon/bin $ ./PixyMon
PixyMon
File Action Help
Pixy detected.
> runprogArg 8 1
response: 0 (0x0)

图 3-18

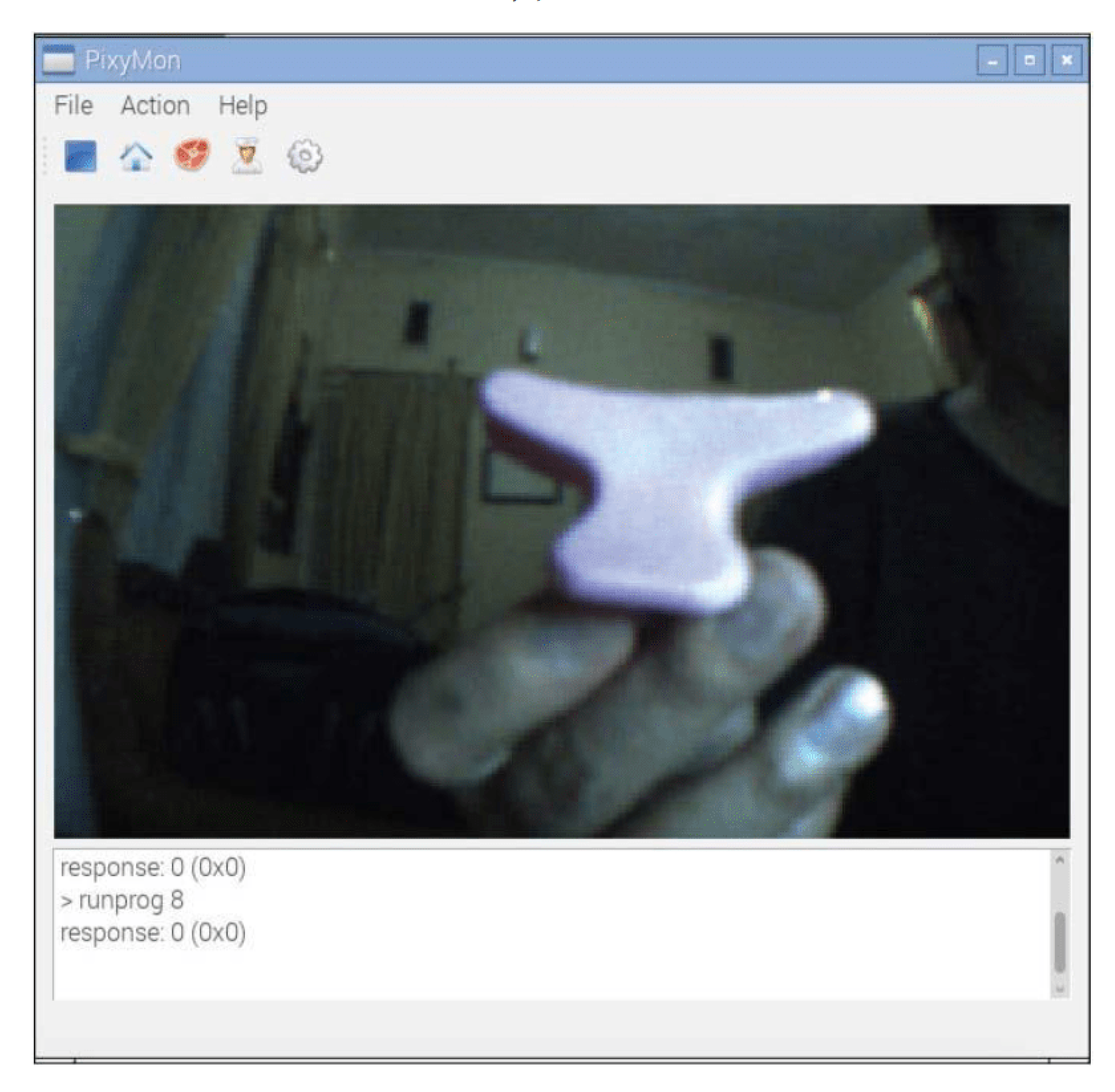

PixyMon
File Action Help
response: 0 (0x0)
> runprog 8
response: 0 (0x0)

图 3-19

（2）在摄像头前展示目标物体，然后在 PixyMon 中选择菜单 Action | Set signature 1，如图 3-20 所示。

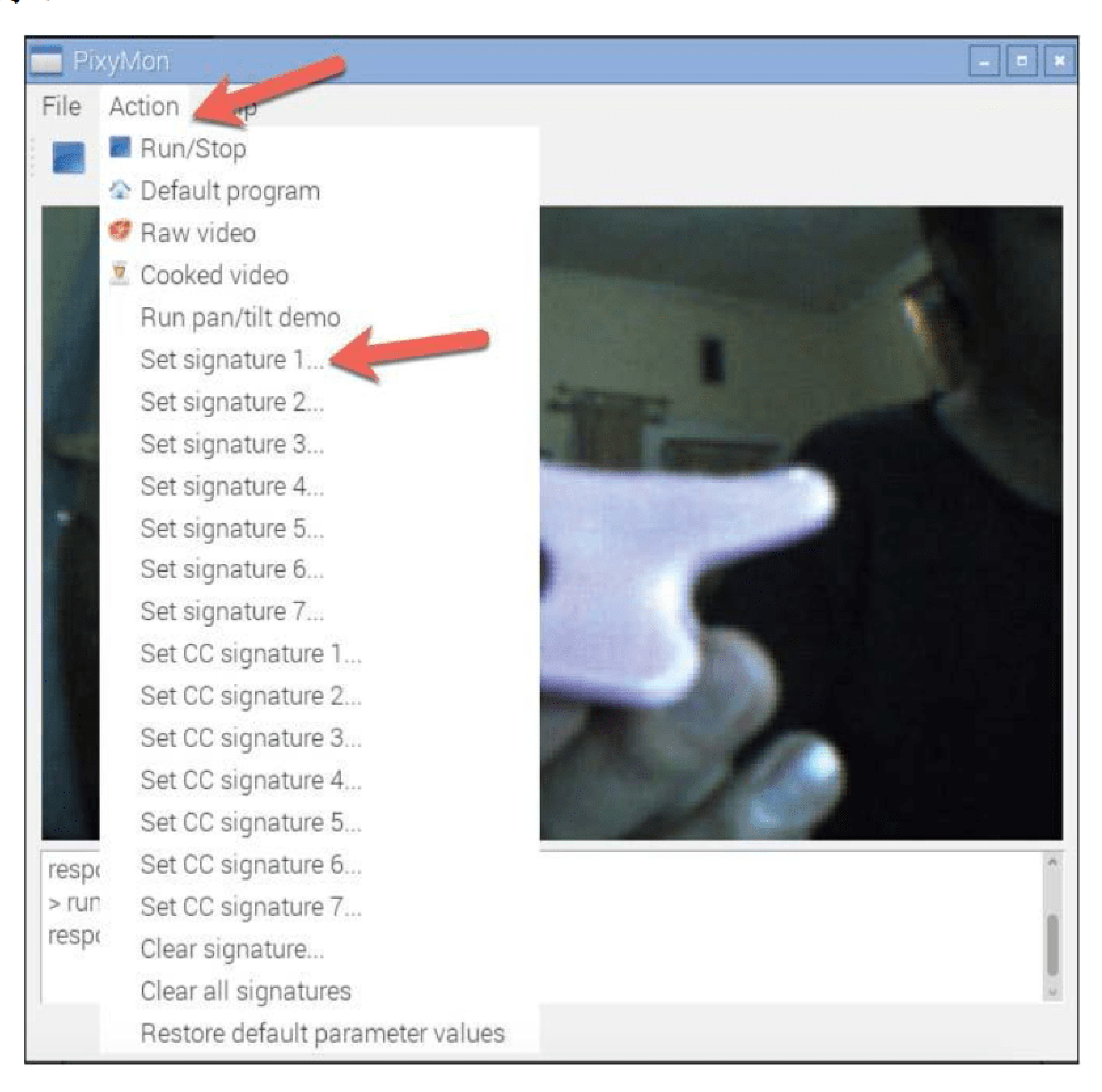


图 3-20

（3）PixyMon 会让图像静止，然后可以用鼠标框出目标物体的范围，如图 3-21 所示。

（4）PixyMon 会保存数据签名，然后就可以跟踪你的物体了，尝试移动目标物体。

利用 Pan/Tilt 模块跟踪物体

如果 Pixy CMUcam5 已经安装了 Pan/Tilt 模块，可以通过 Pan/Tilt 实现一个跟踪物体的例子。

（1）利用已经注册的签名，选择 PixyMon 应用的菜单 Action | Run Pan/Tilt demo，激活 Pan/Tilt，如图 3-22 所示。

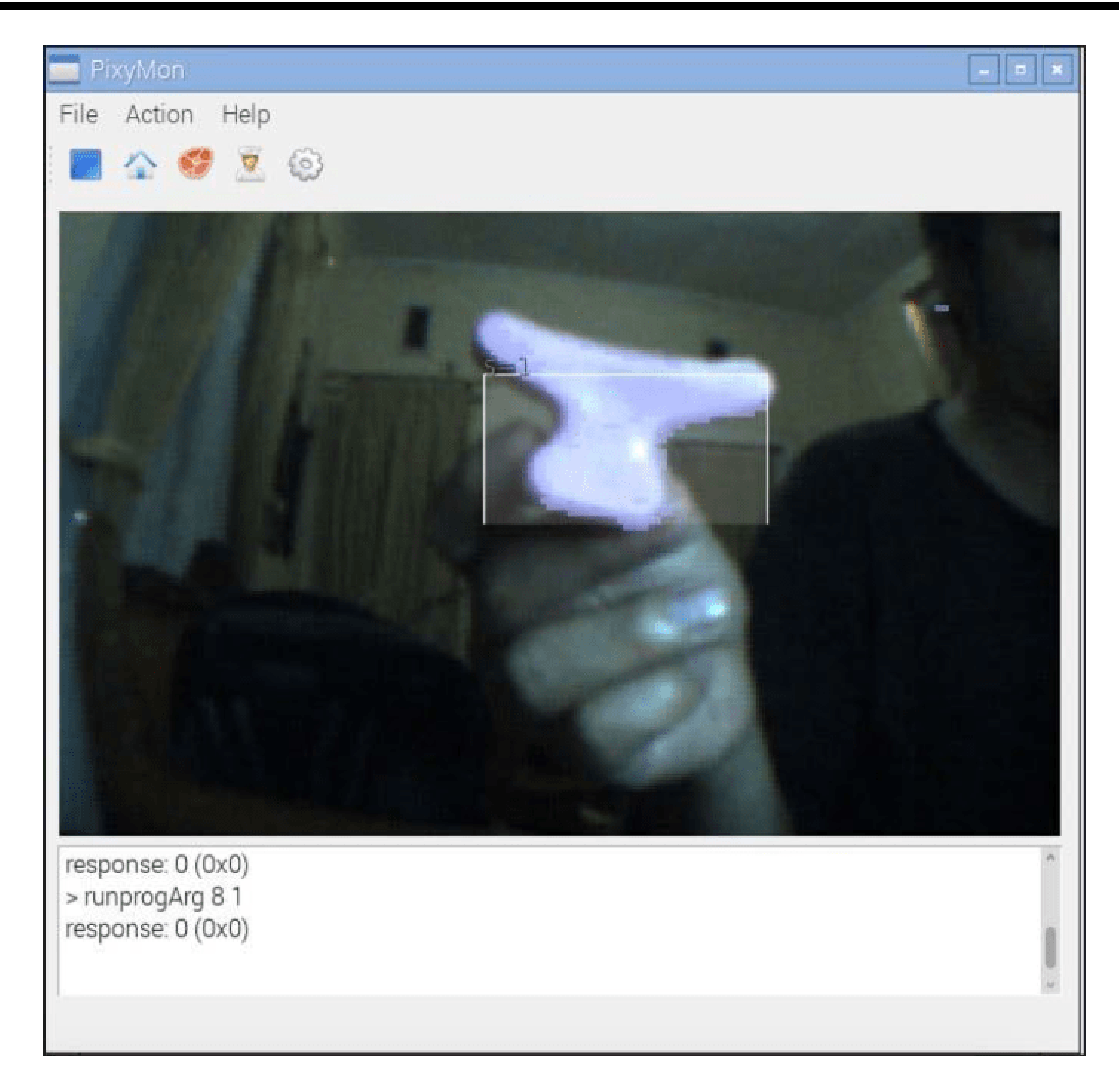

PixyMon
File Action Help
s=1
response: 0 (0x0)
> runprogArg 8 1
response: 0 (0x0)

图 3-21

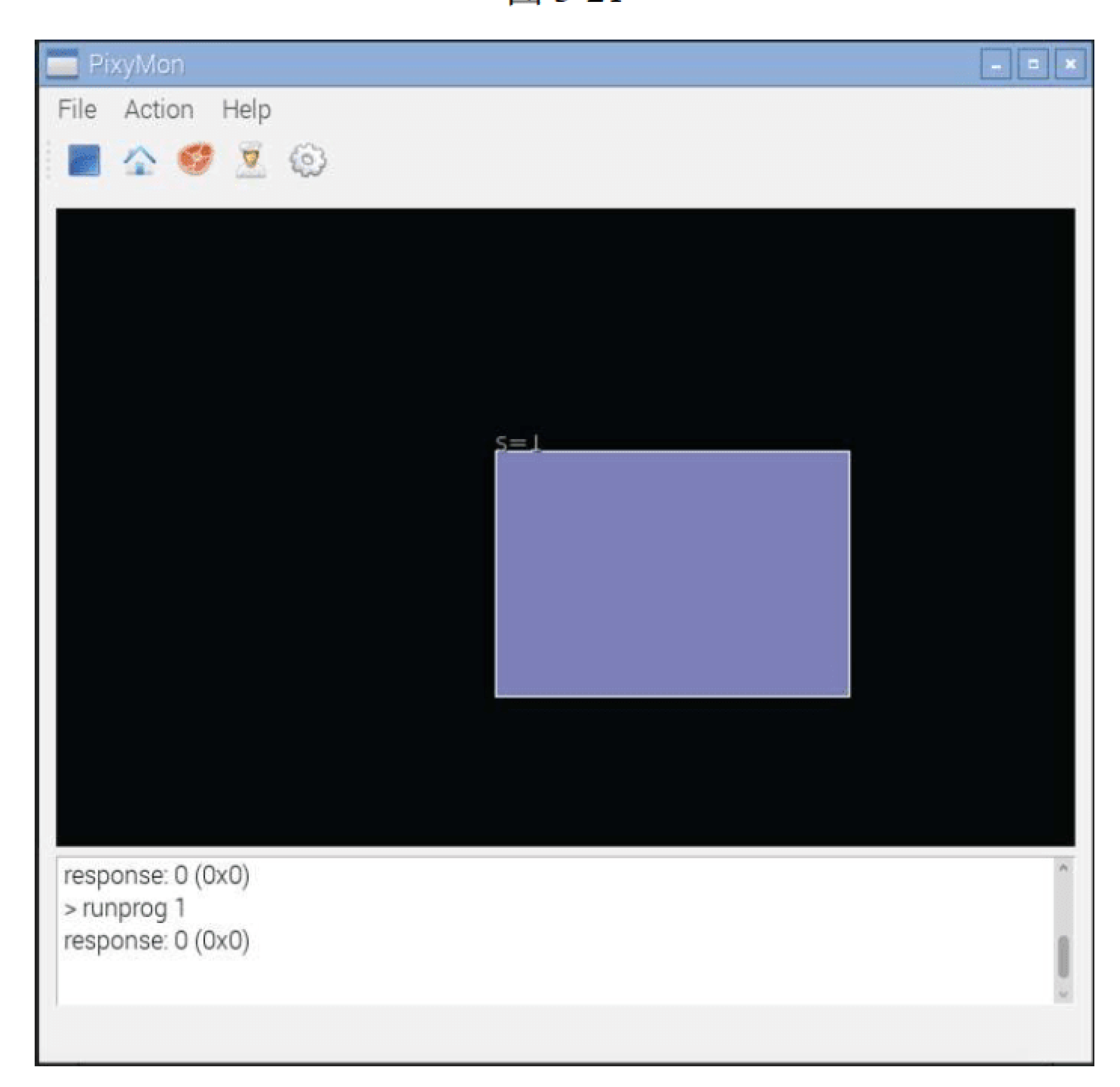

PixyMon
File Action Help
s=1
response: 0 (0x0)
> runprog 1
response: 0 (0x0)

图 3-22

（2）尝试移动目标物体。Pan/Tilt 模块会跟着目标物体的位置移动。

运行 Python 应用

利用同样的注册签名，可以得到一个签名位置。这里使用 Python 写的示例程序。

在文件夹<pixy_codes>/build/libpixyusb_swig/下找到文件 get_blocks.py，接着添加这个文件：

```
$ python get_blocks.py
```

如果成功，程序将获得签名的位置。可以看到程序如图 3-23 所示的输出。

```
agusk — pi@raspberrypi: ~/pixy/pixy/build/libpixyusb_swig — ssh pi@192.168....

pi@raspberrypi:~/pixy/pixy/scripts $ ls
build_hello_pixy.sh          build_pantilt_c_demo.sh       install_libpixyusb.sh
build_libpixyusb.sh          build_pantilt_python_demo.sh  pack_pixymon_src.sh
build_libpixyusb_swig.sh     build_pixymon_src.sh
pi@raspberrypi:~/pixy/pixy/scripts $ cd ..
pi@raspberrypi:~/pixy/pixy $ cd build/
libpixyusb/      libpixyusb_swig/ pixymon/
pi@raspberrypi:~/pixy/pixy $ cd build/libpixyusb_swig/
pi@raspberrypi:~/pixy/pixy/build/libpixyusb_swig $ ls
build  get_blocks.py  pixy.i  pixy.py  _pixy.so  pixy_wrap.cxx  setup.py  src
pi@raspberrypi:~/pixy/pixy/build/libpixyusb_swig $ python get_blocks.py
Pixy Python SWIG Example -- Get Blocks
frame    0:
[BLOCK_TYPE=0 SIG=1 X=309 Y=198 WIDTH=   9 HEIGHT=   2]
frame    1:
[BLOCK_TYPE=0 SIG=1 X=309 Y=198 WIDTH=   9 HEIGHT=   2]
frame    2:
[BLOCK_TYPE=0 SIG=1 X=309 Y=198 WIDTH=   9 HEIGHT=   2]
frame    3:
[BLOCK_TYPE=0 SIG=1 X=309 Y=198 WIDTH=   9 HEIGHT=   2]
```

图 3-23

下一步呢？

可以修改程序，用 Raspberry Pi 板控制 Pixy CMUcam5 模块。

清除所有签名

如果已经完成了所有实验并且不想再使用签名数据，可以通过选择 PixyMon 应用的菜单 Action | Clear all signatures 来清除它们。

总　　结

之前已经利用 OpenCV 学习了一些基本的机器视觉知识，也探索并练习了用 Python 来使用 OpenCV。

在最后的部分，在 Raspberry Pi 板上部署了机器视觉功能来进行人脸识别和目标物体跟踪。

在下一章中，将学习如何使用机器学习的知识制作一个自动车。

引　用

下面是一些推荐的书籍和网站，可以了解更多关于本章的内容。

1. Richard Szeliski. Computer Vision: *Algorithms and Applications*, Springer. 2011.

2. P. Viola and M. Jones, *Rapid object detection using a boosted cascade of simple features*, Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, 2001, pp. I-511-I-518 vol. 1.

3. OpenCV library, http://opencv.org.

第 4 章　制作自动机器车

本章将一起探索如何通过集成一些传感设备和驱动设备来制作一台不需要人为干涉的机器小车，同时也会学习如何用计算机来控制和导航这辆机器车。

本章分为以下几个部分：

- 自动系统介绍
- 移动机器人介绍
- 制作机器车
- 使用 Pololu Zumo Arduino
- 用计算机控制机器车
- 添加用于导航的 GPS 模块
- 地图引擎平台介绍
- 构建基于导航的车载 GPS
- 制作自动机器车

自动系统介绍

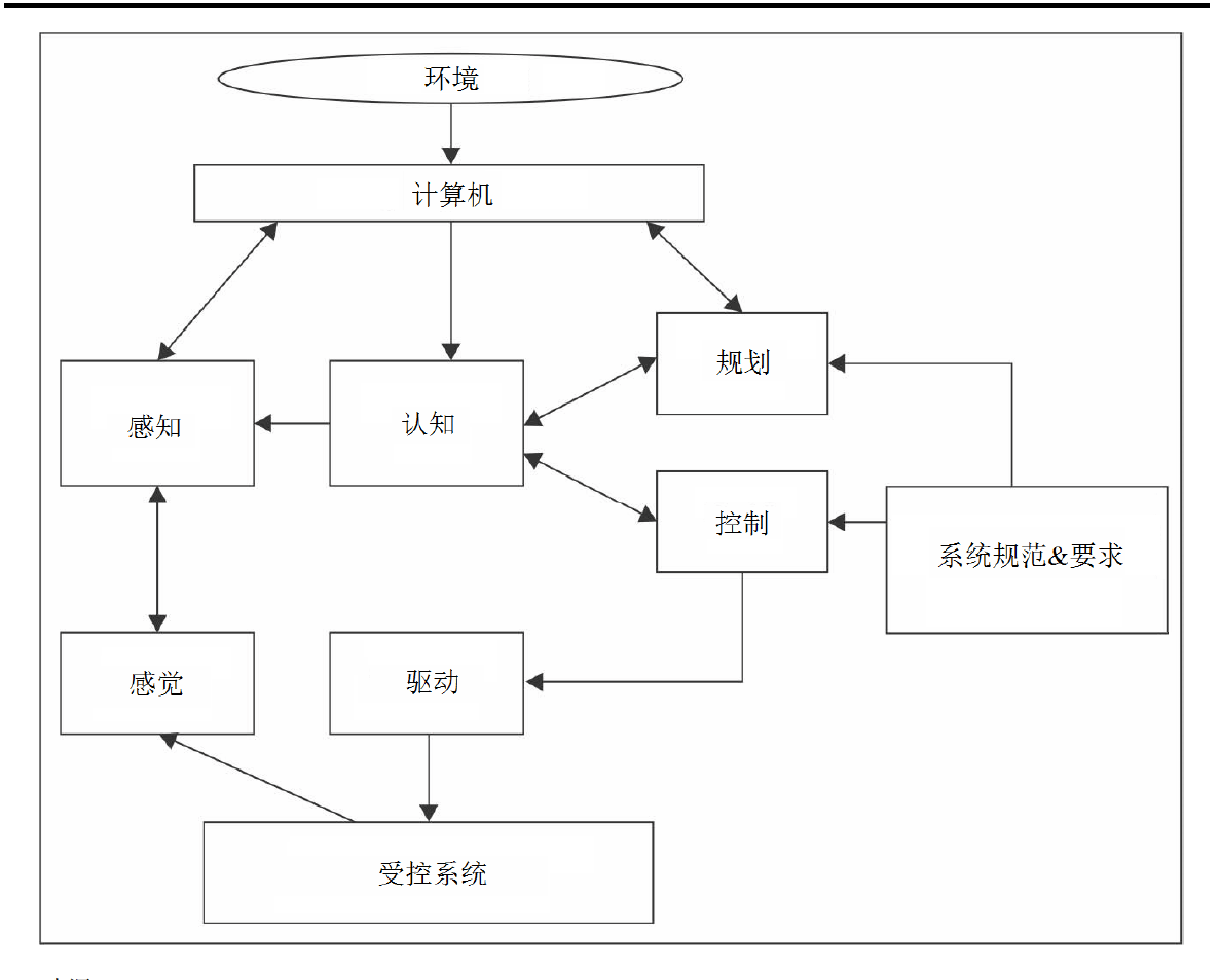
自动系统是指能够通过自学习决策或行动的系统。在传统系统中，为系统准备一些要做的任务序列，然后系统根据这个序列依次执行。但是自动系统不一样，自动系统自己学习该执行什么操作。

在机器人研究领域，自动机器人系统有两个基本问题：

- 路径与运动规划问题
- 运动控制问题

路径与运动规划问题指的是从一个位置到另一个位置，机器人选取什么路线。 机器人在移动的过程中，既可以依靠地图也可以不依靠。运动控制问题是指机器人是怎么移动的：从一个位置到另一个位置时，它既可能像走迷宫那样，也可能像走之字形一样。

一般来说，自动系统的架构如图 4-1 所示。



来源：Meystel, A.: Intelligent Control: A Sketch of the Theory, 1. Intelligent and Robotic Systems, Vol. 2, 1989, pp. 97-107.

图 4-1

从图 4-1 可知，搭建一个自动系统有如下 6 个关键部分：

- 认知部分
- 感知部分
- 规划部分
- 控制部分
- 传感部分
- 传动部分

本章将学习如何实现自动系统的关键部分。这里采用现有的机器人平台和 Arduino 板来部署自动机器人。

介绍移动机器人

移动机器人是指利用发动机来具备移动能力的机器人。一般来说，可以使用图 4-2 描述的 5 个关键部分搭建移动机器人。

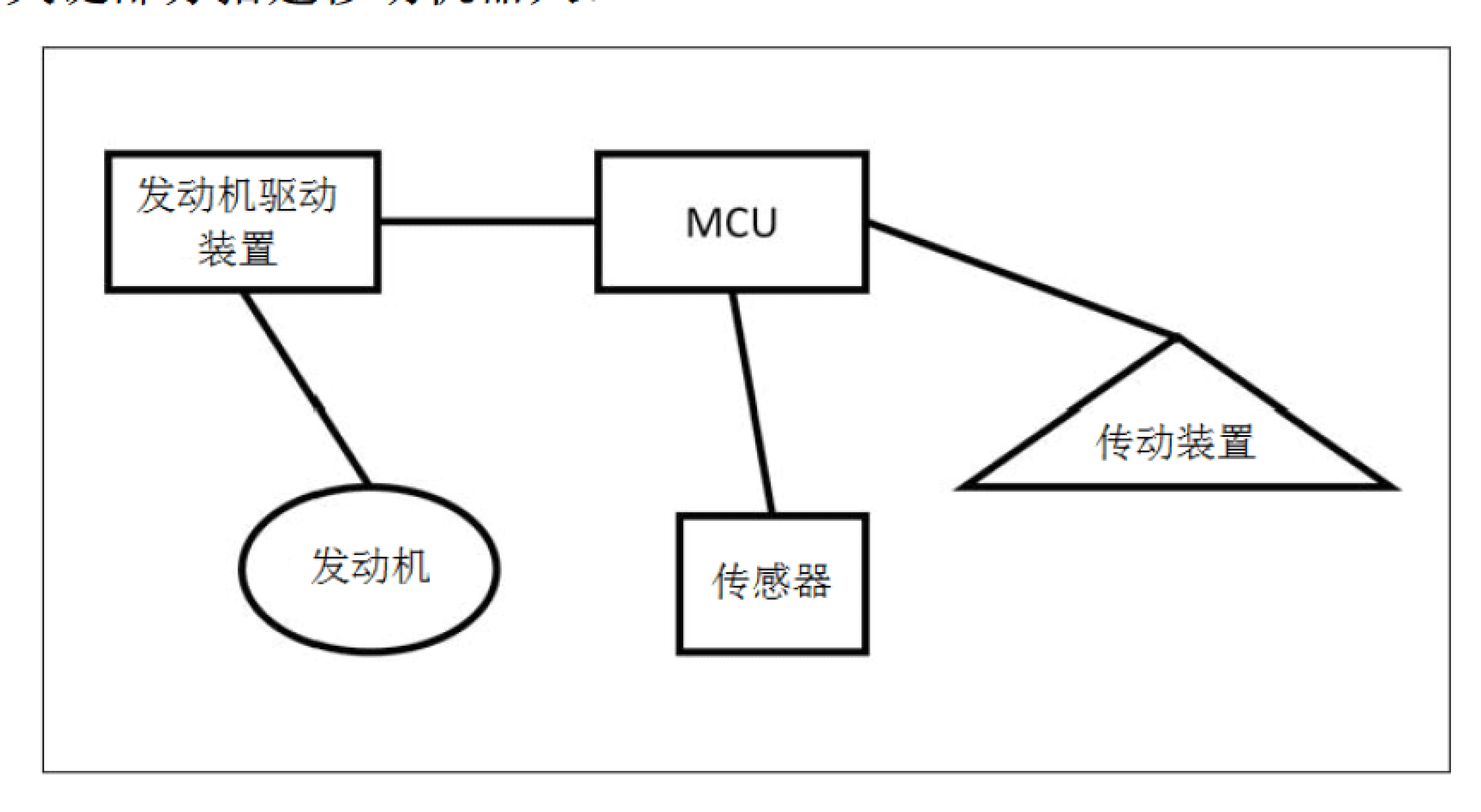


图 4-2

MCU（微处理中央单元）是机器人的控制中心。可以选择 Arduino、Intel Edison、BeagleBone Black/Green，或者是 Raspberry Pi board（树莓派）。每种 MCU 有自己的特性和编程方式。用户需要根据自己的需求选择不同大小和重量的 MCU。

可以根据机器人的移动类型选择适合的发动机模型。由于一些 MCUs 缺少 PWM/Analog 输出，为了控制发动机，需要一个发动机驱动装置。原因是，发动机通常都需要较高的电压，因此如果直接连接 MCU 和发动机会导致 MCU 损坏，而发动机驱动装置就起到了调压器的作用，使得发动机电压得到控制。

需要一些传感模块来搜集机器人所处的外界环境的信息。传感模块具备将外界的物理输入转化为数字信息的能力。可以在 MCU 上添加各种各样的传感模块。不同的传感器在相同的环境中输出不同的信息，例如有的输出温度，有的输出湿度。摄像机也可以扮演光学传感器的角色，用来捕捉机器人所在的环境。

最后，机器人也要具备和外界环境交互的能力。这个可以通过传动装置实现。举个例子，LED 灯就是一个非常简单的可以传达特定信息的传动装置。设想这样一个场景，一个搭载着气体检测传感器的机器人突然监测一个危险气体，于是另一个机器人立刻亮起红灯作为危险信号。也可以在机器人平台上使用传动装置，不过要注意是否有足够的

电量给传动装置提供电能。

搭建机器车

本章将学习如何搭建一个机器车。因为有许多不同的选择，下文将给出一些笔者认为需要考虑到的内容。

- 目的：为什么想要搭建一个机器车呢？是为了娱乐、研究，还是为了完成一个专业项目？
- 微处理中央单元：需要想好在为机器车编写程序时用到什么级别的复杂度。用 Arduino 板足够吗？是不是还需要更加复杂一点的 MCU board，如 Raspberry Pi？
- 电池：这个非常重要。可以根据预期的电量使用时间选择合适的电池。
- 传感器和传动装置：请使用适量合适的传感器和传动装置。因为这些使用的越多，意味着将需要更多的电量，同时这也会非常影响机器人的重量。

除此之外，还应该草拟一份关于机器人性能的清单。

下面将回顾两种不同的机器人平台：DIY 平台和集成平台。

DIY 机器人平台

DIY（Do-It-Yourself）是指自己干，因此这个平台需要搭建者具备很多的创造性和动手能力。另外，一些机器人供应商往往还会销售所需的工具配件。在这个平台上，搭建者对搭建机器人完全单独负责，甚至包括焊接电子组件。

SparkFun Inventor's Kit for RedBot，网址为 https://www.sparkfun.com/products/12649。图 4-3 所示为这个 DIY 机器人平台的示例。

SparkFun Inventor's Kit for RedBot 是一个完整的机器人，需要自己组装所有的部分。这里面包括一个基于 Arduino 的 MCU 板，可以非常容易地在上面编程。

笔者还在一个中国制造商那里发现了一个非常便宜的 DIY 机器人平台，叫作 DIY Smart Motor Robot Car，来自 Banggood（http://www.banggood.com/DIY-Smart-Motor-Robot-Car-Chassis-Battery-Box-Kit-Speed-Encoder-For-Arduino-p-1044541.html），可以从 dx.com、eBay、AliExpress 和阿里巴巴购买到。图 4-4 所示为 DIY Smart Motor Robot Car 的一个示例。



图 4-3



图 4-4

不过 DIY Smart Motor Robot Car 套件不包含 MCU 板，所以需要单独购买一个，如 Arduino 或者 Raspberry Pi。当拿到之后，可以按照说明来搭建机器车。

笔者相信在读者所在的当地肯定有许多 DIY 套件。期待你的创造性！

集成的机器人平台

如果你比较懒，或者你不想自己手动焊接这些电子部分，可以购买一个集成的机器人套件，如图 4-5 所示，这样就只需要专注于编程而不用从零开始。一些机器人供应商也买集成的套件。根据想用的编程模型选择机器人平台，如希望在 Arduino 平台上编程，那么可以使用 Pololu Zumo Arduino（https://www.pololu.com/product/2510）。

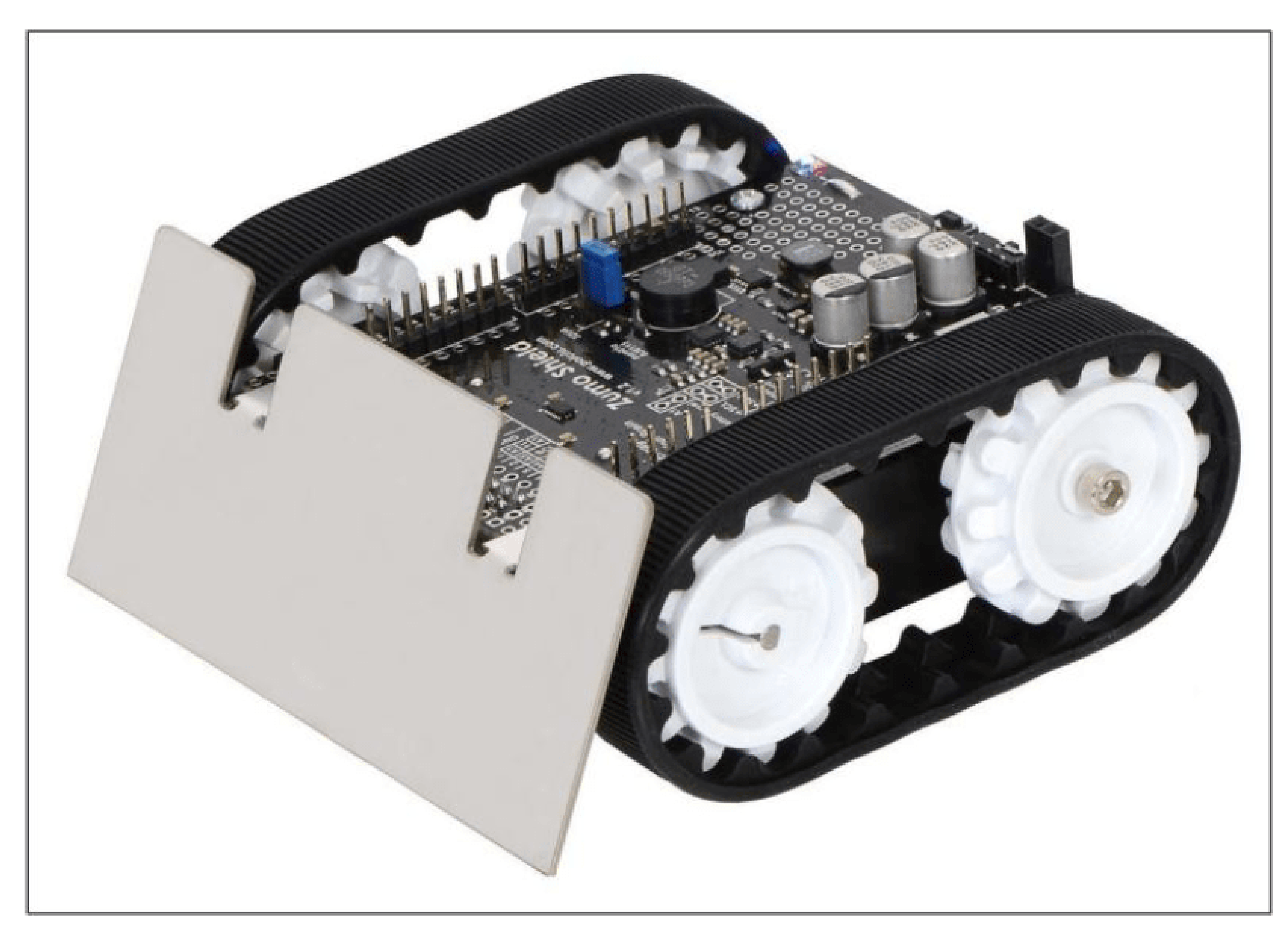


图 4-5

Zumo Robot 被设计成一个 Arduino shield，支持 Arduino Uno 模型。还可以使用 Arduino Leonardo、Arduino 101 和 Arduino Zero。请确保 Arduino pin 电压为 5V。

还可以用来自 Pololu 的 Zumo 32U4 Robot 作为另一个选择，如图 4-6 所示。在 https://www.pololu.com/product/3125 中可以找到。这里不需要 Arduino 板，因为 Zumo 32U4 Robot 中已经集成了一个。只需要编写一个 Sketch 程序，然后部署到板子上即可。

另一个可供选择的是来自 Makeblock 的 Makeblock mBot V1.1，如图 4-7 所示，网址是 http://makeblock.com/mbot-v1-1-stem-educational-robot-kit。这个套件是基于 Arduino Uno 的完整的机器人。在通信模块 Makeblock mBot V1.1 提供蓝牙和无线电上两个不同的选择，可以按照需求进行购买。

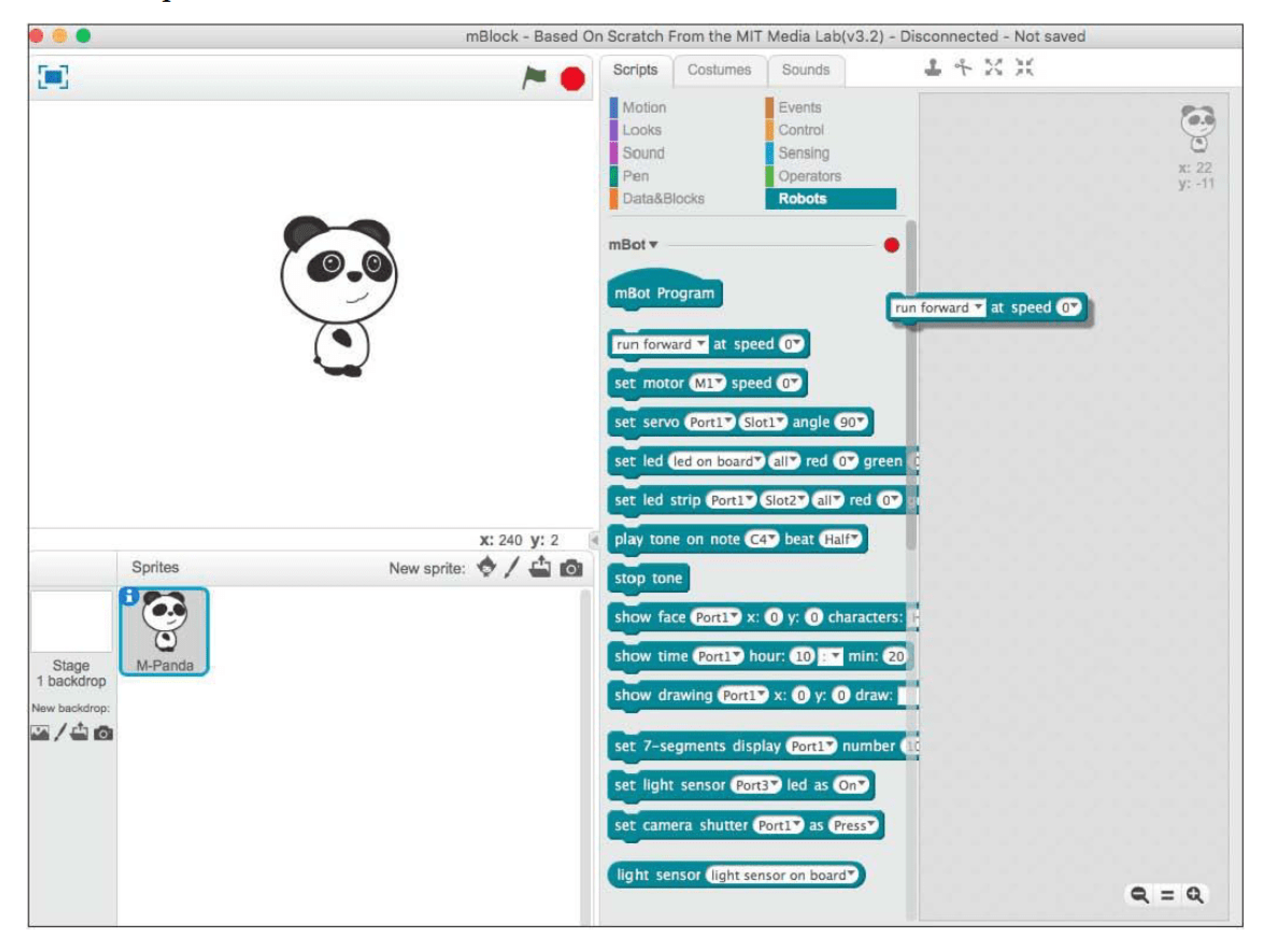


图 4-6



图 4-7

为了给 Makeblock mBot 编程，Makeblock 提供了一个开发工具，叫作 mBlock，是基于 Scratch（https://scratch.mit.edu）的客户定制版。可以通过单击和拖曳来编程，如图 4-8 所示。



来源：http://learn.makeblock.com/en/getting-started-programming-with-mblock/

图 4-8

使用 Pololu Zumo robot for Arduino

本节将制作一个简单的能够移动避开障碍物的机器人。规则是如果机器人遇到一个障碍物就向左转。为了实现这个功能，用一个 Ultrasonic 模块来检测障碍物。

笔者通常使用 HC-SR04 作为 Ultrasonic 模块。它很便宜，可以在 www.dx.com、www.banggood.com 和 AliExpress 上买到。HC-SR04 模块如图 4-9 所示。

HC-SR04 模块提供 4 个 pins：VCC、GND、Trigger 和 Echo。可以连接 Trigger 和 Echo pins 到 Arduino 的数字 I/O。但是怎么把 HC-SR04 模块连接到 Pololu Zumo robot 呢？

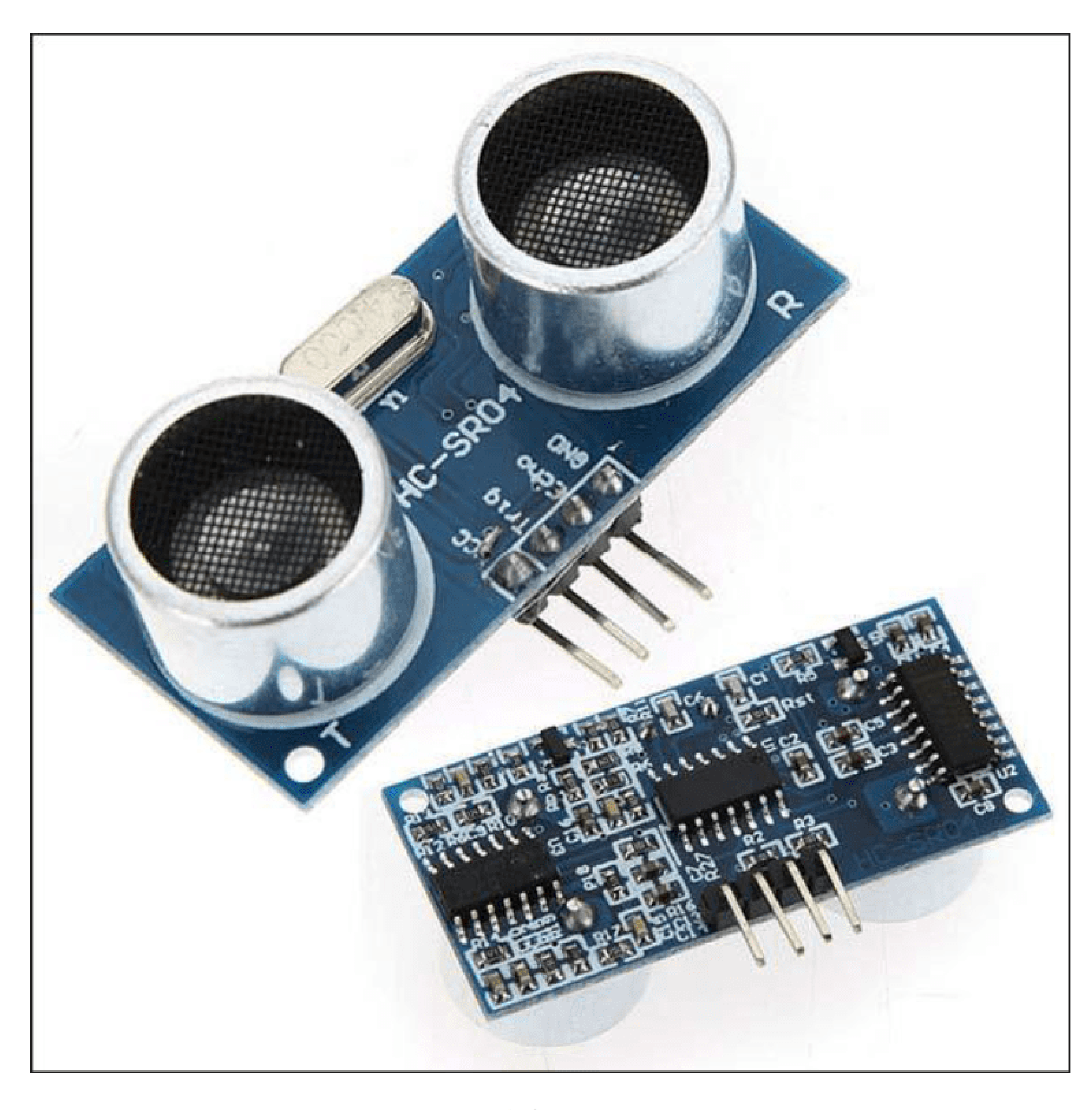


图 4-9

根据 Pololu 的用户手册，可以通过 Arduino 数字 I/O 的 4、11、5、2 这 4 个 pin 连接。连接 HC-SR04 模块最方便的方法是使用前面扩展的 pins，具体如图 4-10 所示。

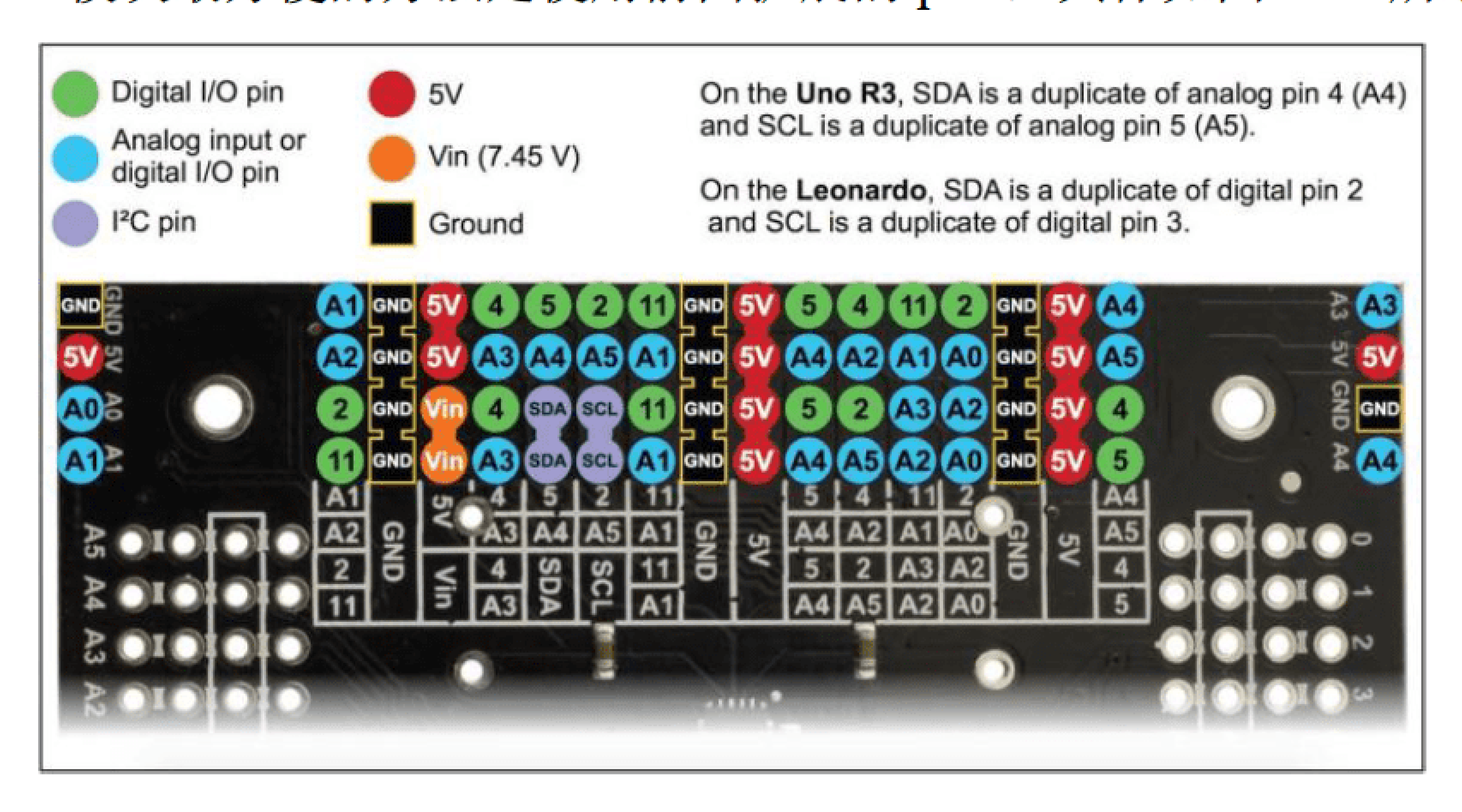


图 4-10

在这个例子中，连接 Trigger 和 Echo pins 到 Arduino 数字 I/O 的 pins 2 和 4。Ultrasonic

模块的 VCC 和 GND pins 分别对应连接到 Arduino 的 5V 和 GND pins。图 4-11 所示是连接示例。

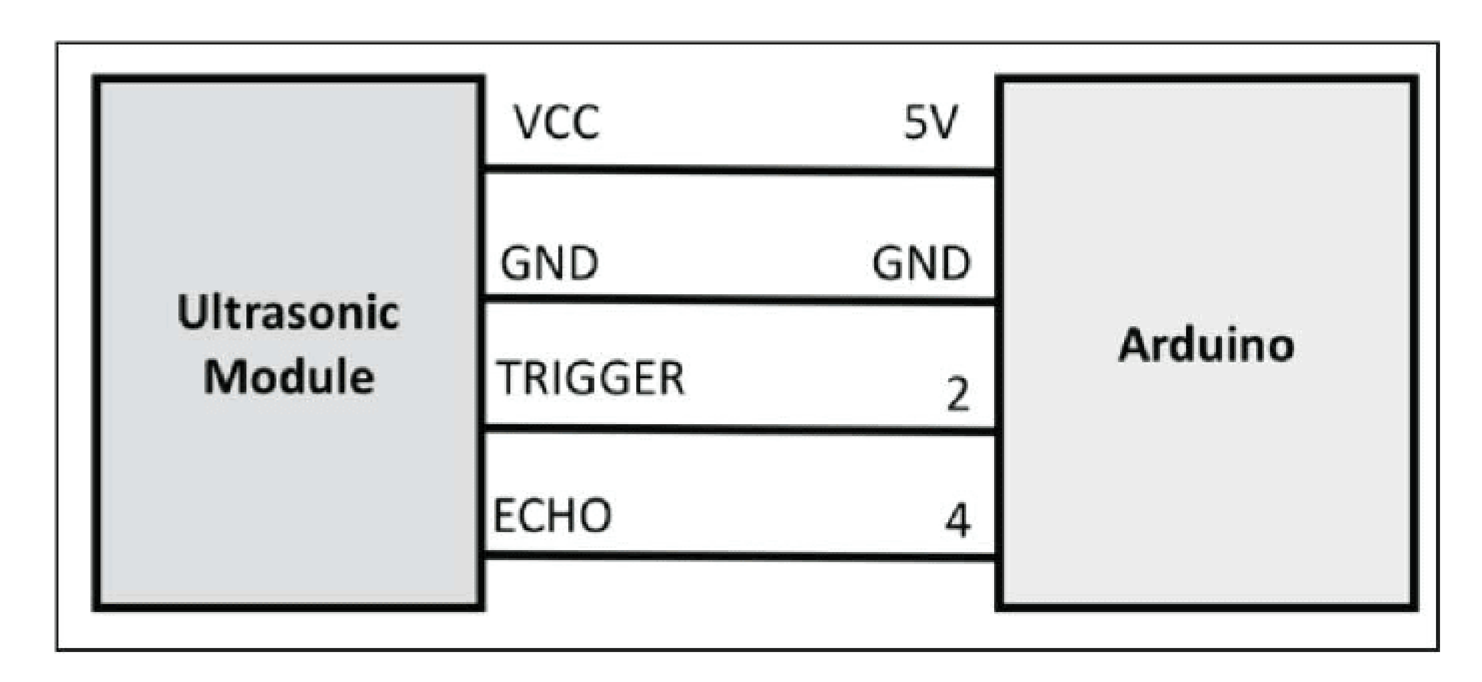


图 4-11

此外，还可以尝试用排针把 Ultrasonic 模块焊接到 Pololu Zumo robot 的屏蔽板上，但要确保使用的是数字 pins 4、11、5 和 2，因为这些 pins Pololu Zumo robot 屏蔽板不会用到。

图 4-12 所示是接线的一个例子。

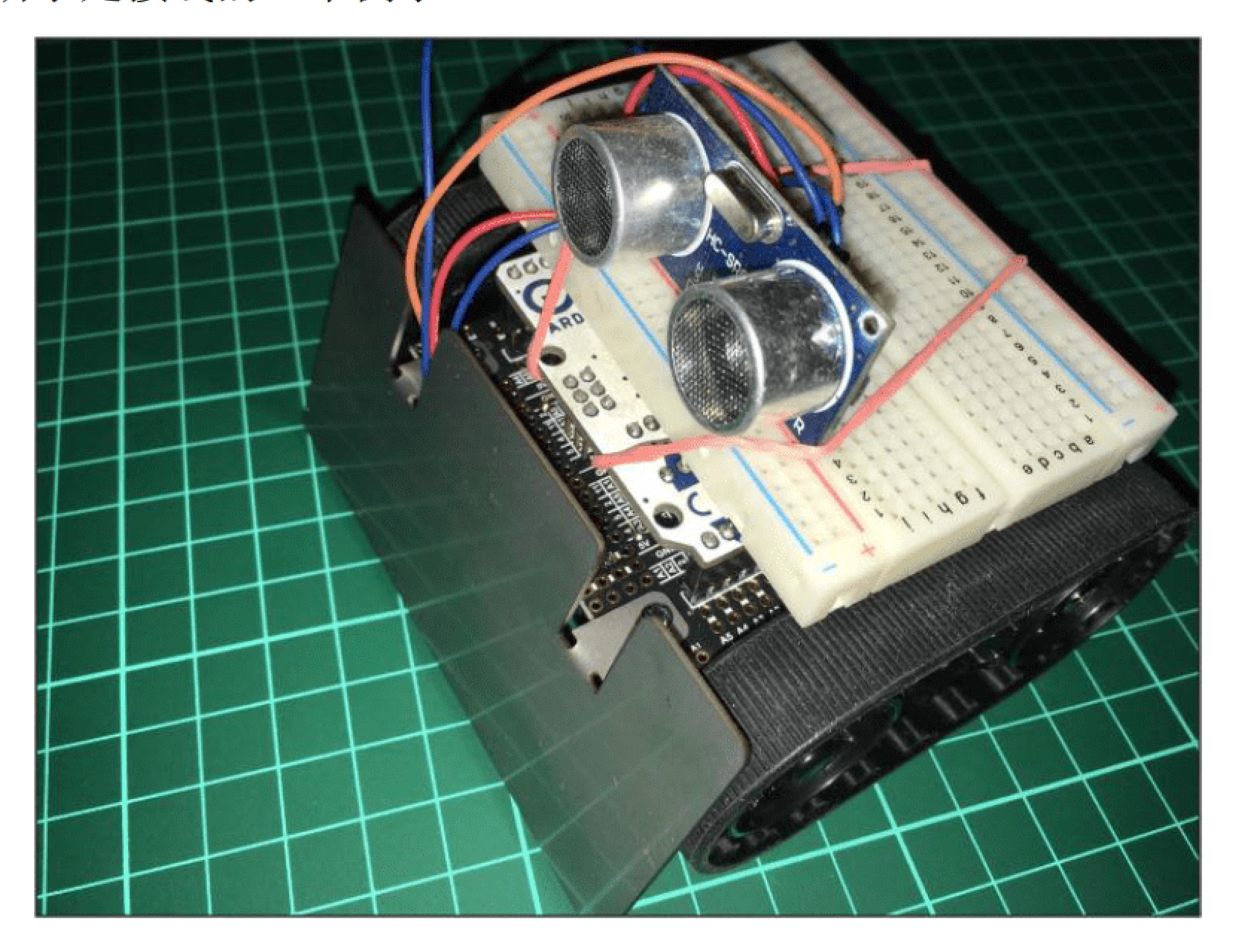


图 4-12

我们将在 Sketch 程序里使用 NewPing 库。可以从 http://playground.arduino.cc/Code/NewPing 下载，然后把这个库部署到 Arduino libraries 文件夹。之后就可以开始编写 Sketch 程序。

打开 Arduino IDE，然后编写以下代码：

```
#include <NewPing.h>
#include <ZumoMotors.h>

#define TRIGGER_PIN  2
#define ECHO_PIN     4
#define MAX_DISTANCE 600

NewPingsonar(TRIGGER PIN, ECHO PIN, MAX DISTANCE);
ZumoMotors motors;
long duration, distance;

void setup() {
pinMode(13, OUTPUT);
pinMode(TRIGGER_PIN, OUTPUT);
pinMode(ECHO_PIN, INPUT);
Serial.begin(9600);
}

void loop() {

digitalWrite(TRIGGER_PIN, LOW);
delayMicroseconds(2);

digitalWrite(TRIGGER_PIN, HIGH);
delayMicroseconds(10);

digitalWrite(TRIGGER_PIN, LOW);
duration = pulseIn(ECHO_PIN, HIGH);

  //根据声音的速度计算距离（cm）
distance = duration/58.2;

Serial.println(distance);
motors.setRightSpeed(100);
motors.setLeftSpeed(100);
delay(200);

if(distance <= 20) {

digitalWrite(13, HIGH);  //换一条路

motors.setLeftSpeed(-300);
```

```
motors.setRightSpeed(100);

delay(200);

}else{
digitalWrite(13, LOW);

motors.setLeftSpeed(100);
motors.setRightSpeed(100);
delay(200);
  }
}
```

把程序保存为 ch04_01，然后上传程序到 Arduino 板。

现在可以打开 Pololu Zumo robot，把机器人放在房间的角落来测试它是否可以躲避障碍物。

如何运行

首先，定义好 Ultrasonic 模块的 pins：

```
#define TRIGGER_PIN  2
#define ECHO_PIN     4
#define MAX DISTANCE 600
NewPingsonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
```

在 setup()函数里，把 TRIGGER_PIN 作为输出，ECHO_PIN 作为输入：

```
pinMode(TRIGGER_PIN, OUTPUT);
pinMode(ECHO_PIN, INPUT);
```

然后，通过 TRIGGER_PIN 广播信号。之后，计算在 ECHO_PIN 上的持续时间：

```
digitalWrite(TRIGGER PIN, LOW);
delayMicroseconds(2);

digitalWrite(TRIGGER PIN, HIGH);
delayMicroseconds(10);
digitalWrite(TRIGGER_PIN, LOW);
duration = pulseIn(ECHO_PIN, HIGH);
```

将 Ultrasonic 模块的持续时间通过除以 58.2 转化为距离，这个值是基于声音的速度得到的：

```
distance = duration/58.2;
```

如果这个距离小于 20cm，机器人应该向左转弯。可以修改机器人该怎么应对的算法：

```
if(distance <= 20) {
digitalWrite(13, HIGH);  //换一个方向
motors.setLeftSpeed(-300);
motors.setRightSpeed(100);
delay(200);
}else{
digitalWrite(13, LOW);

motors.setLeftSpeed(100);
motors.setRightSpeed(100);
delay(200);
   }
```

用计算机控制机器车

可以用计算机来控制机器车。这意味着可以通过给机器发送命令来执行动作。在计算机和机器车上各需要一个通信模块来完成通信。

本节将在 Pololu Zumo robot 和计算机之间实现通信，模式如图 4-13 所示。通常使用蓝牙模块作为通信的无线栈 。

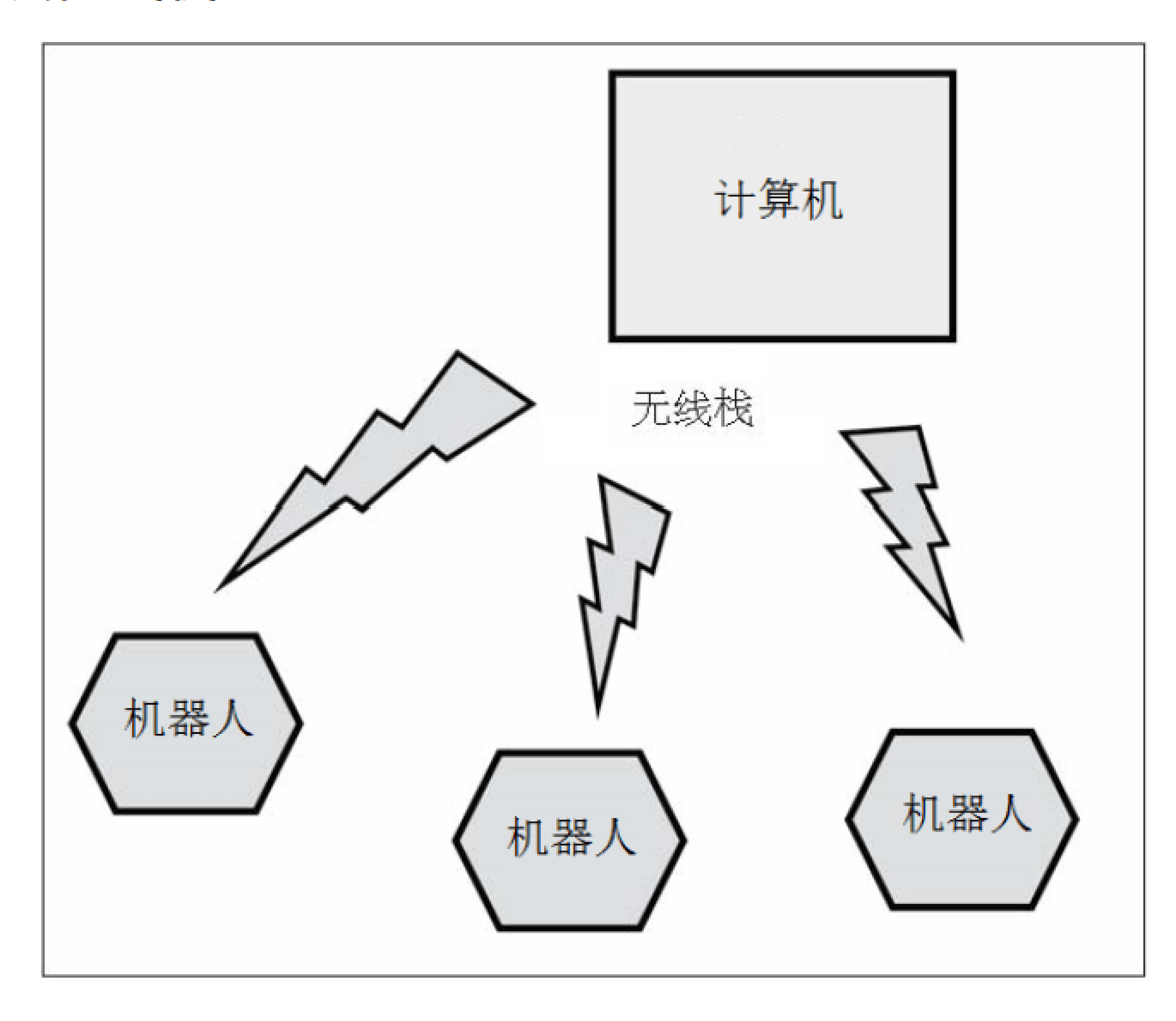


图 4-13

具体来说，这里使用蓝牙 HC-06，一个蓝牙接受装置，这样能直接通过 UART 协议通信。HC-06 模块价格很实惠，可以在 www.banggood.com、eBay、www.dx.com 和 AliExpress 买到。

HC-06 模块有如下 4 个输出 pins：

- VCC
- GND
- Rx
- Tx

一种蓝牙 HC-06 模块如图 4-14 所示。

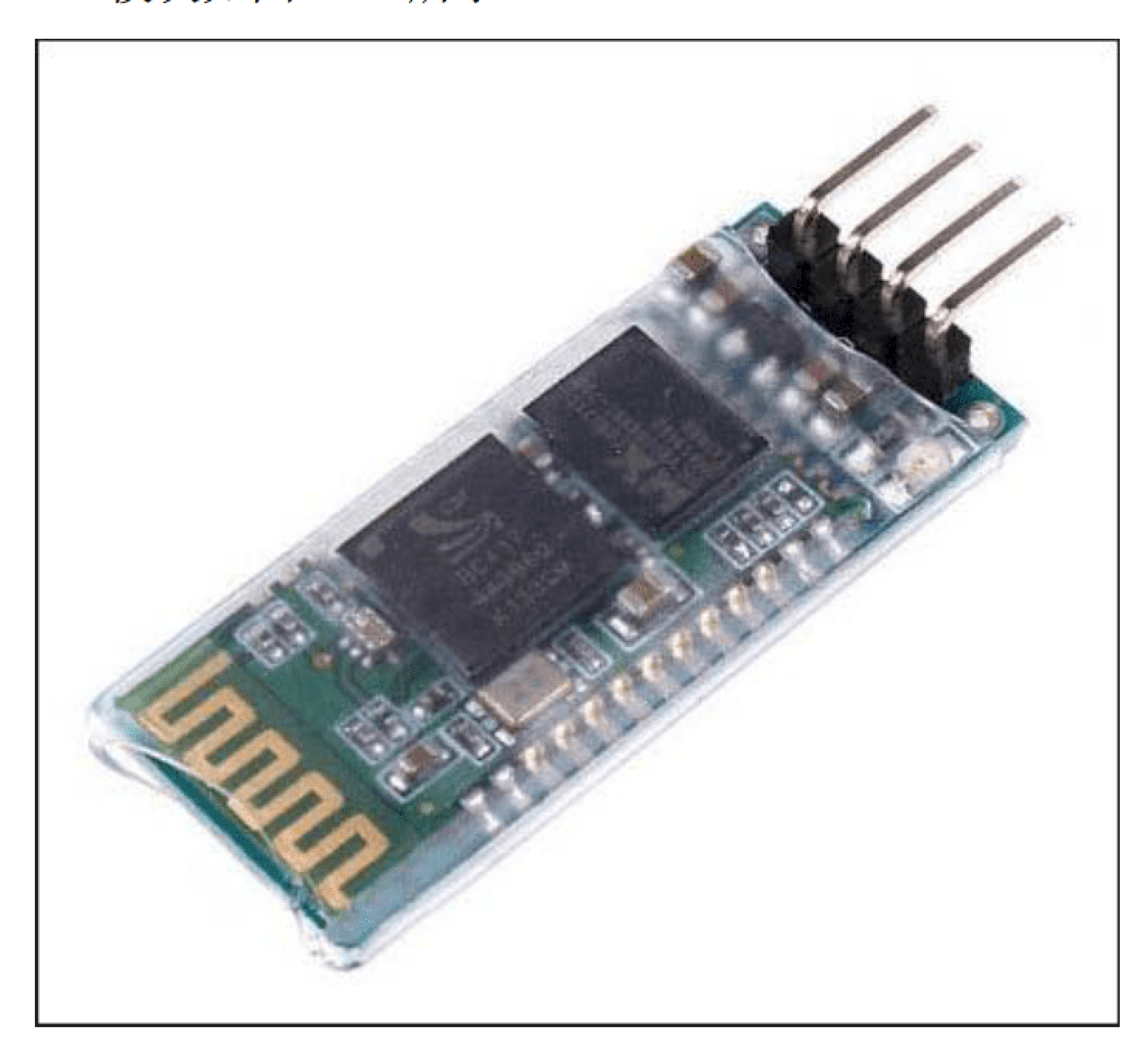


图 4-14

这里使用 SoftwareSerial 库（https://www.arduino.cc/en/Reference/SoftwareSerial）和蓝牙 HC-06 通信。这个库支持 AVR MCU 模块。但是如果使用基于 Arduino 的 ARM MCU，如 Arduino 101 和 Arduino Zero，就不能使用 SoftwareSerial 库。

对于 Arduino Leonardo，不是所有的数字 pins 能被用在 SoftwareSerial 库。可以在 SoftwareSerial 库的网站上确认。

在这个例子中，将使用被添加到 Pololu Zumo robot 作为屏蔽板的 Arduino Uno R3。

下面定义 Arduino 数字 pins 2 和 4 作为 Rx 和 Tx，这样可以把 Pololu Zumo robot 通过前伸缩板连接到蓝牙 HC-06 模块。

下面是一个连线例子：

- ❑ 蓝牙 HC-06 VCC 连接前伸缩板 5V。
- ❑ 蓝牙 HC-06 GND 连接前伸缩板 GND。
- ❑ 蓝牙 HC-06 Rx 连接前伸缩板上的 pin 4（Tx）。
- ❑ 蓝牙 HC-06 Tx 连接前伸缩板上的 pin 2（Rx）。

现在可以给 Pololu Zumo robot 编写 Sketch 程序。打开 Arduino IDE 并编写以下代码：

```
#include <ZumoMotors.h>
#include <SoftwareSerial.h>

//D2   >>>  Rx, D4   >>>Tx
SoftwareSerialbluetooth(2, 4); //RX, TX
charval;
ZumoMotors motors;

void setup() {
Serial.begin(9600);
pinMode(13, OUTPUT);
bluetooth.begin(9600);
Serial.println("Bluetooth On..");
}

void loop() {
if(bluetooth.available()){
digitalWrite(13, HIGH);
val = bluetooth.read();

Serial.println(val);
if(val == 'l' ) {
motors.setLeftSpeed(-300);
motors.setRightSpeed(100);
Serial.println("turn left");
   }
if(val == 'r' ) {
motors.setRightSpeed(-300);
motors.setLeftSpeed(100);
```

```
Serial.println("turn right");
    }
if(val == 'f' ) {
motors.setLeftSpeed(100);
motors.setRightSpeed(100);
Serial.println("forward");
    }

digitalWrite(13, LOW);

  }

delay(200);
}
```

把代码保存为 ch04_02，然后上传到 Arduino 板。

在计算机上匹配蓝牙 HC-06。这里用的是苹果计算机匹配蓝牙 HC-06。如图 4-15 所示，在蓝牙设置里，应该可以看到蓝牙 HC-06 在蓝牙列表中。

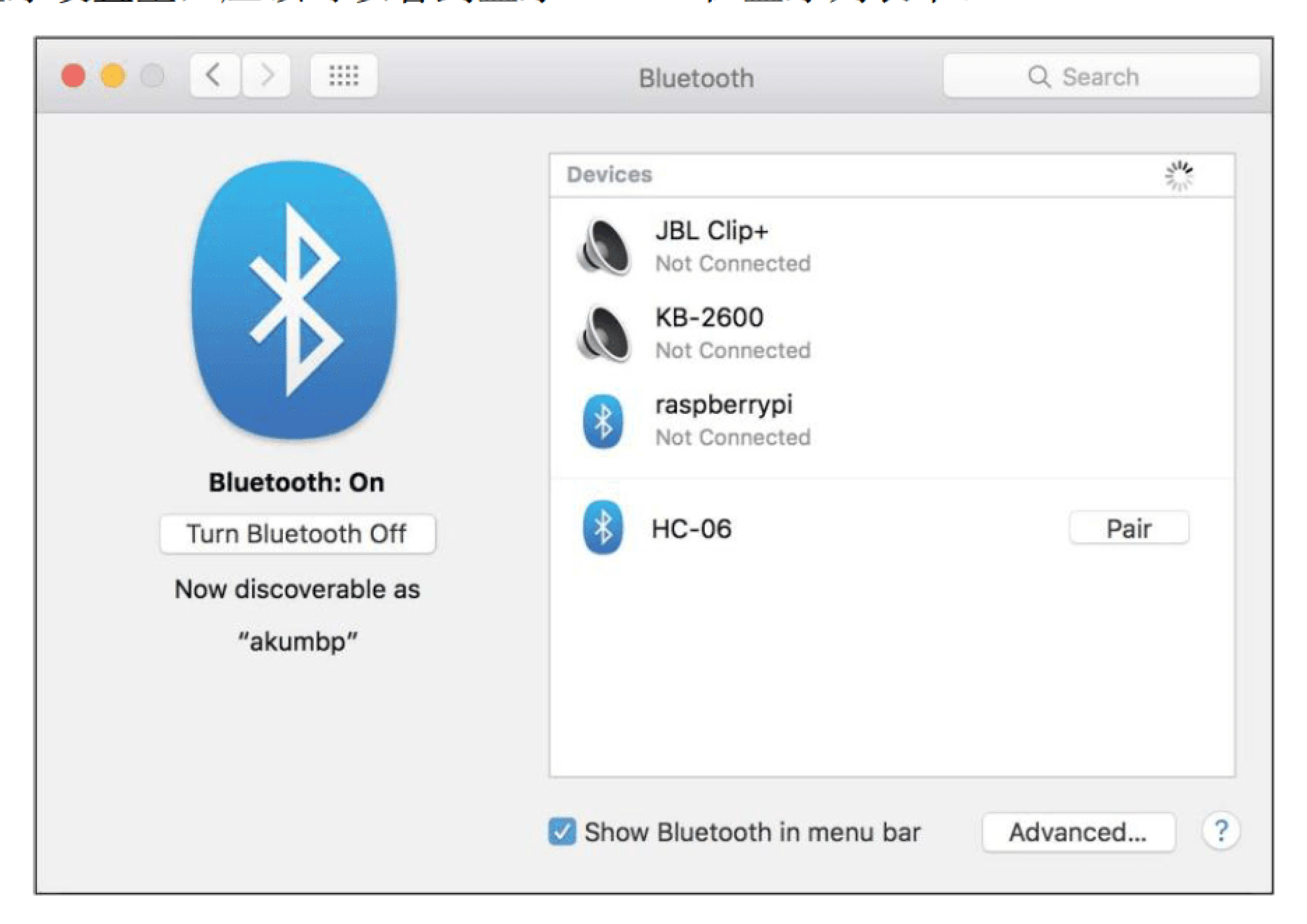


图 4-15

开始匹配蓝牙 HC-06。默认的配对密码是 1234，如图 4-16 所示。

如果看到 HC-06 连接到计算机，那么就表示配对成功了，如图 4-17 所示。

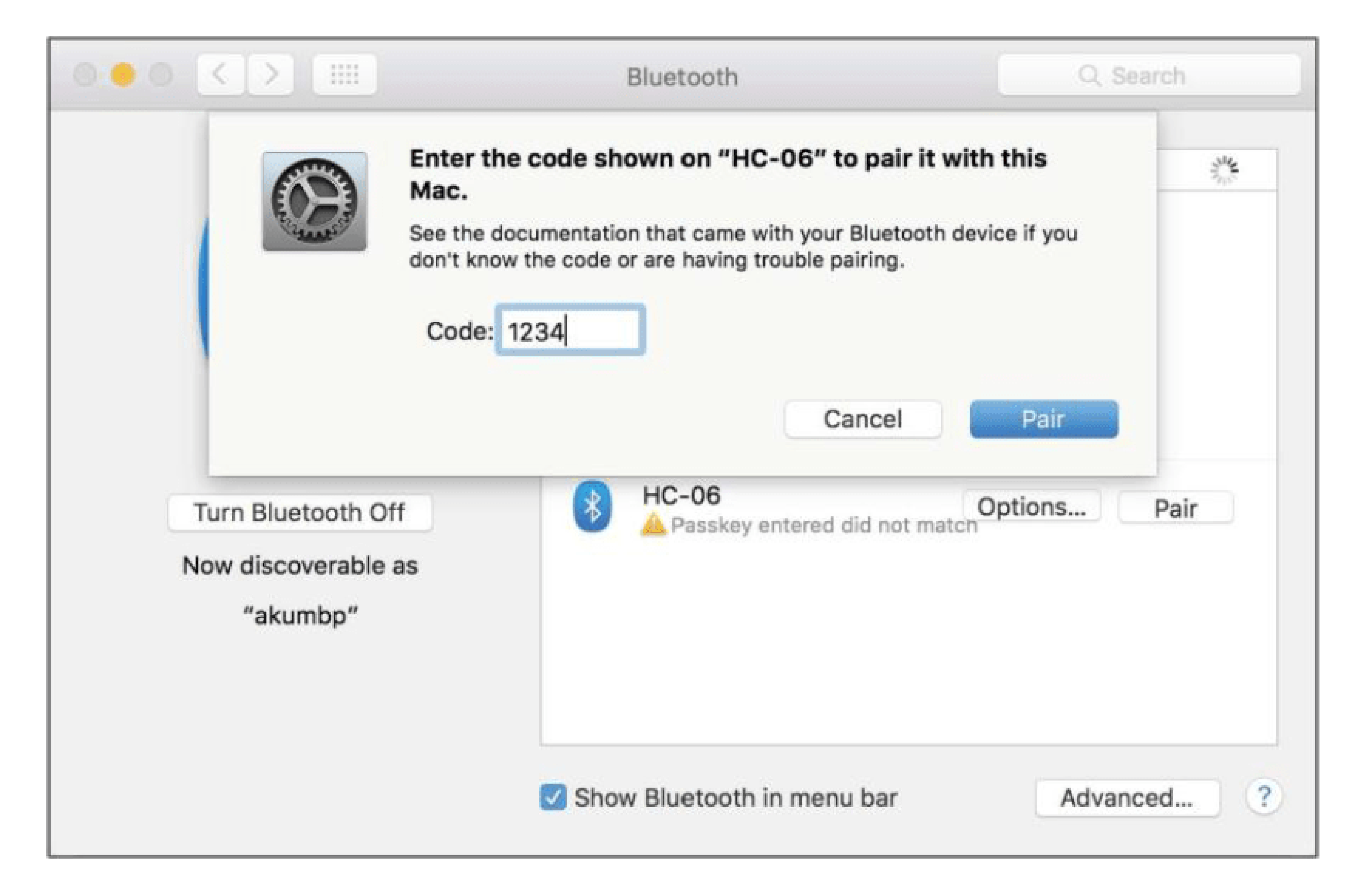


图 4-16

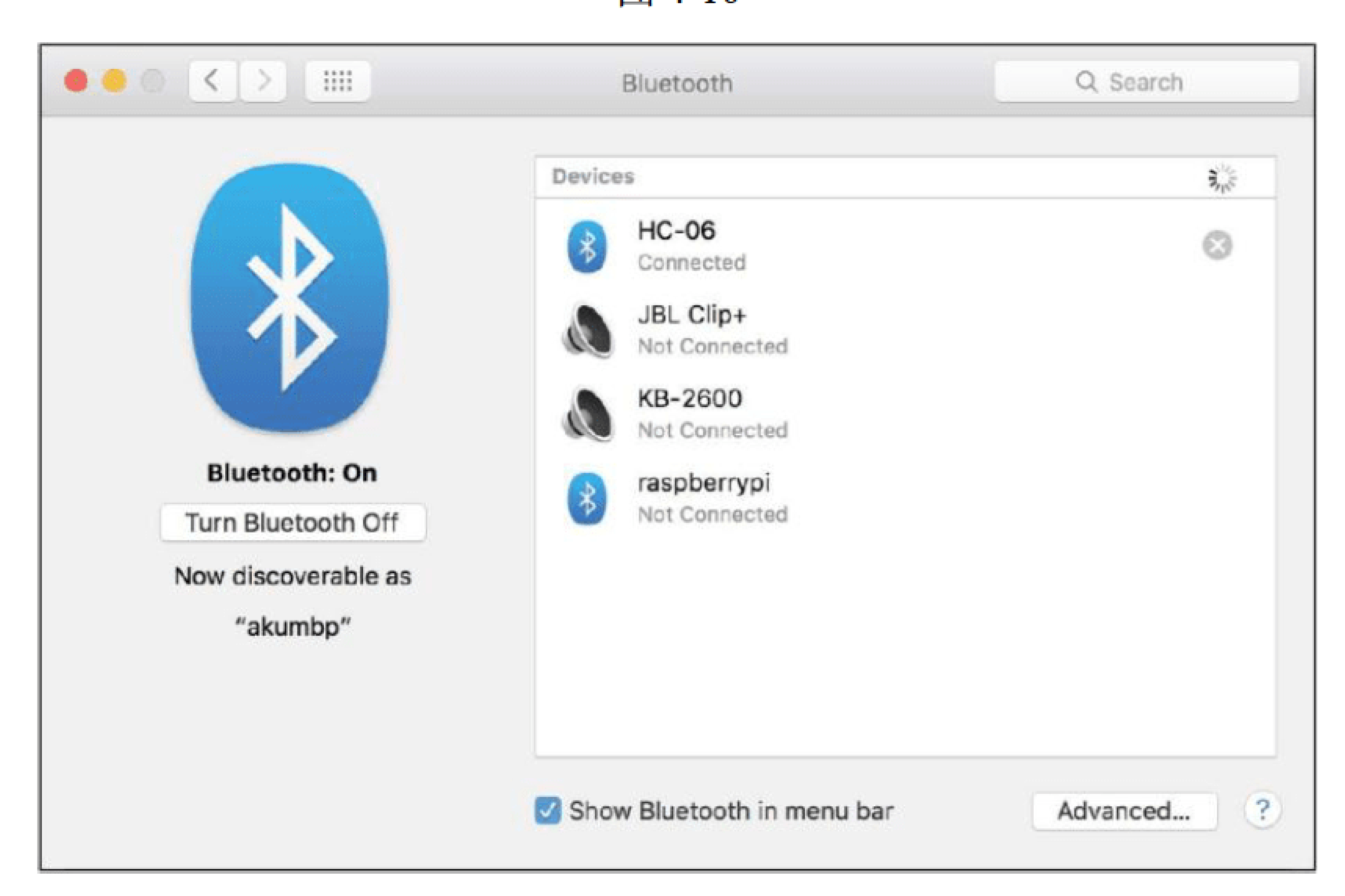


图 4-17

现在可以验证 HC-06 的串行口。在命令行里输入如下命令：

```
$ ls /dev/cu*
```

下面查看 HC-06 串行口。我的计算机上检测到的是/dev/cu.HC-06-DevB，如图 4-18 所示。稍后将在程序里用到。

现在可以编写 Python 程序来访问 HC-06 的端口。这里使用 pyserial 库 https://pypi.python.org/pypi/pyserial 来访问。

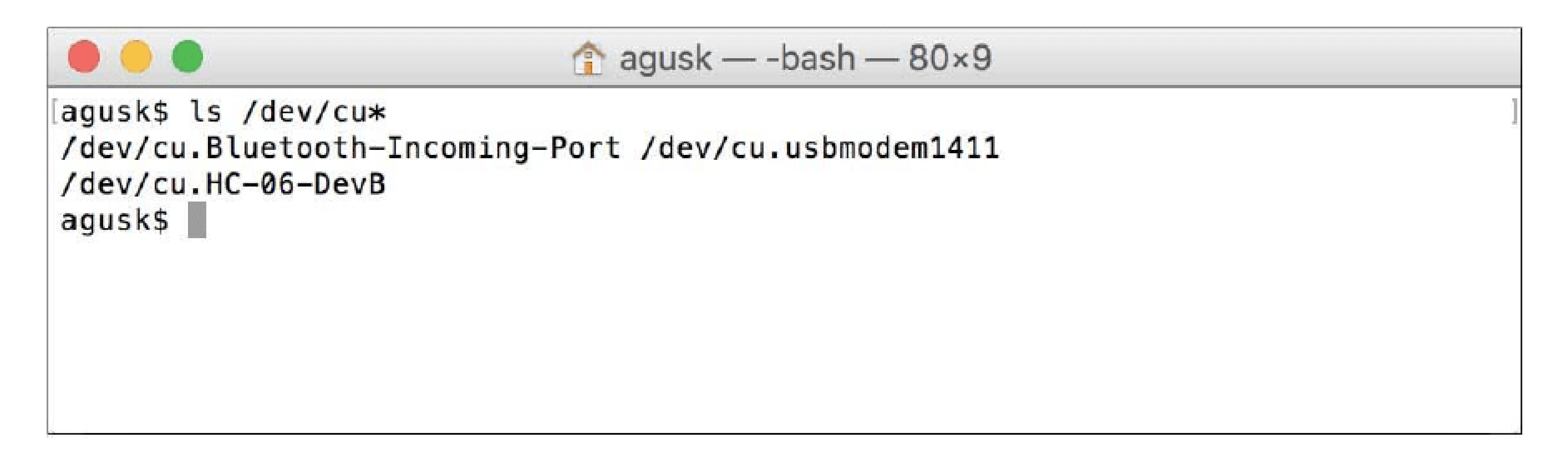


图 4-18

可以用如下命令来安装这个库。

```
$ pip install pyserial
```

有可能需要在前面添加 sudo。

下一步就是编写 Python 程序。输入如下代码：

```
import serial

serial_hc06 = '/dev/cu.HC-06-DevB'
counter = 0

print('open ', serial_hc06)
hc06 = serial.Serial(serial_hc06, 9600)

while True:
try:
        # python 3.x
        #c = input('>> ')

        # python 2.7.x
        c = raw input('>> ')

if c == 'q':
break

        hc06.write(c)

except (KeyboardInterrupt, SystemExit):
hc06.close()
raise

print('Exit')
```

可能需要把 serial_hc06 的值修改为蓝牙 HC-06 模块的串行口。把代码保存为 ch04_03.py。然后运行程序：

```
$ python ch04_03.py
```

在命令行里出现“>>”后，可以输入 r、l 和 f，分别代表向右转、向左转和直行。输入 q 退出程序。图 4-19 所示是一个示例。

```
codes — -bash — 80×20
agusk$ python ch04_03.py
('open ', '/dev/cu.HC-06-DevB')
>> r
>> l
>> f
>> g
>> h
>> r
>> f
>> q
Exit
agusk$
```

图 4-19

如果 Arduino 板连接到了计算机，可以通过串口监视器看到 Arduino 串行口的消息，接收到的命令就是从计算机上发送的命令。

图 4-20 所示就是串口监视器一个输出实例。

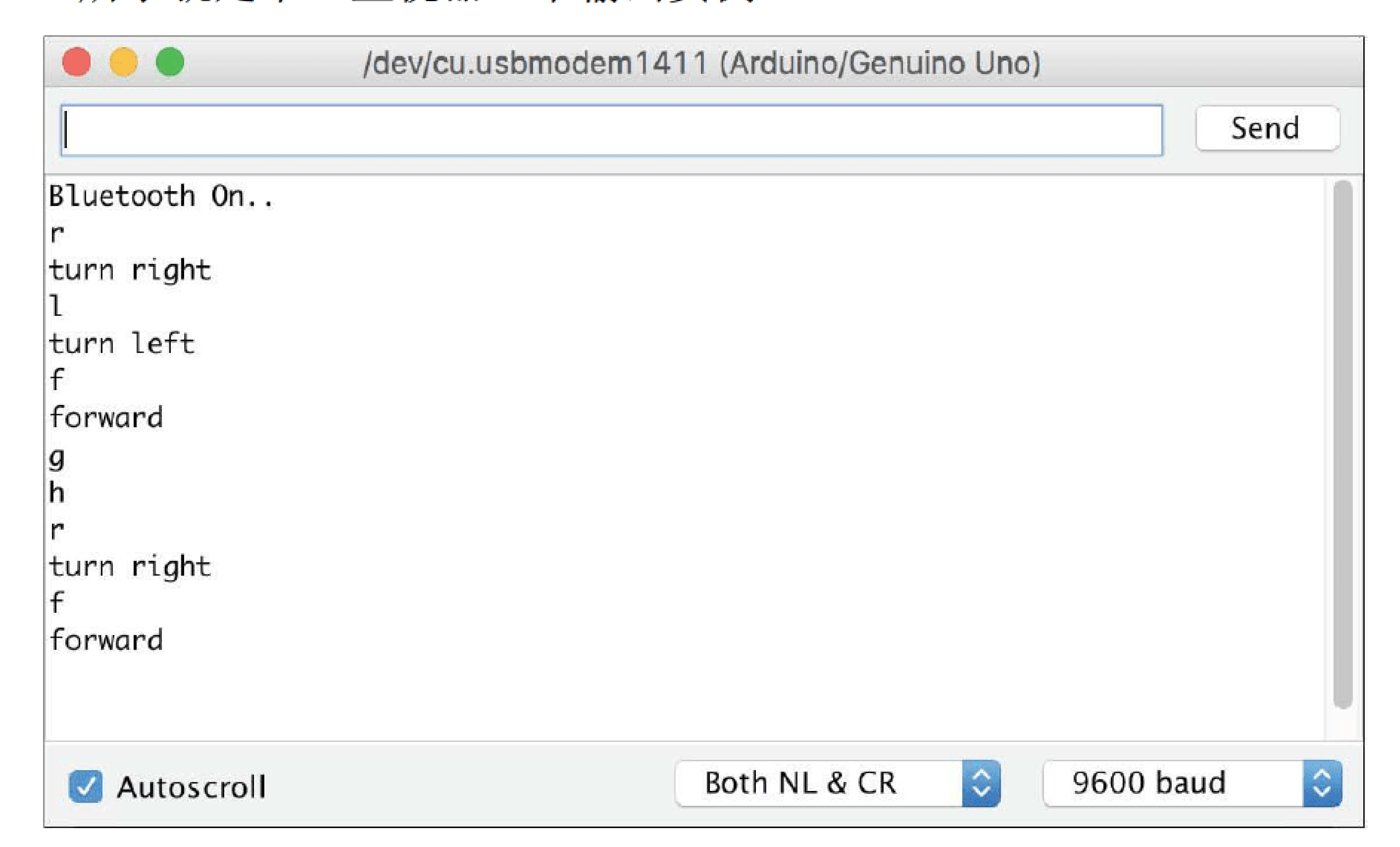


图 4-20

如何运行

程序已经部署到两个设备上了——Arduino 和计算机。添加到 Pololu Zumo robot 的 Arduino 板通过蓝牙接收来自计算机的命令。

在 Sketch 程序里定义发动机和蓝牙模块：

```
SoftwareSerialbluetooth(2, 4); //RX, TX
charval;
ZumoMotors motors;
```

接着 Pololu Zumo robot 将会等待从蓝牙发送过来的命令。如果命令是 r，机器车将会右转，对应的，命令 l 会让机器车左转，接收到命令 f 则会直走：

```
if(bluetooth.available()){
digitalWrite(13, HIGH);
val = bluetooth.read();

Serial.println(val);
if(val == 'l' ) {
motors.setLeftSpeed(-300);
motors.setRightSpeed(100);
Serial.println("turn left");
    }
if(val == 'r' ) {
motors.setRightSpeed(-300);
motors.setLeftSpeed(100);
Serial.println("turn right");
     }
if(val == 'f' ) {
motors.setLeftSpeed(100);
motors.setRightSpeed(100);
Serial.println("forward");
    }
digitalWrite(13, LOW);
  }
```

在计算机端也编写一个 Python 程序，这个程序负责通过串行口发送数据给蓝牙。在

计算机端使用 pyserial 库来访问串行口。

首先定义一个 HC-06 的串行口，可以在 HC-06 和计算机蓝牙连接过之后获得这个串行口。

```
serial_hc06 = '/dev/cu.HC-06-DevB'
counter = 0
print('open ', serial_hc06)
hc06 = serial.Serial(serial_hc06, 9600)
```

串行口打开后，计算机等待用户的输入，一旦获得输入，程序就把它发送给串行口：

```
# python 3.x
#c = input('>> ')

# python 2.7.x
c = raw_input('>> ')
```

用户输入字符 q 退出。

使用 GPS 模块导航

GPS 是一个可以从卫星获得特定位置的模块。GPS 是在室外定位的好方法，一些卫星被用于服务 GPS 模块。

本节将从 GPS 模块获得位置信息，位置信息又会通过蓝牙发送到计算机端。

使用 U-blox NEO-6M 来接收从卫星发来的位置信息。在这个例子里同样还是使用蓝牙模块 HC-06。计算机监听来自机器车的 GPS 数据。GPS 数据就是指经纬度坐标。

可以从 dx.com（http://www.dx.com/p/gps-module-w-ceramic-passive-antenna-for-raspberry-pi-arduino-red-384916）网站上购买 U-blox NEO-6M。这个模块也很便宜，还可以在 Banggood、eBay 和 AliExpress 上购买。

如图 4-21 所示是 U-blox NEO-6M 模块的一个示例。

U-blox NEO-6M 提供 TTL 输出以便获得数据。因为收到的 GPS 是原始数据，所以需要一个 GPS 数据解析器。使用 TinyGPS 库（https://github.com/mikalhart/TinyGPS）来编码 GPS 数据。下载 TinyGPS 代码之后把它放到 Arduino 库中。

下一步是给出连线示例，如图 4-22 所示。

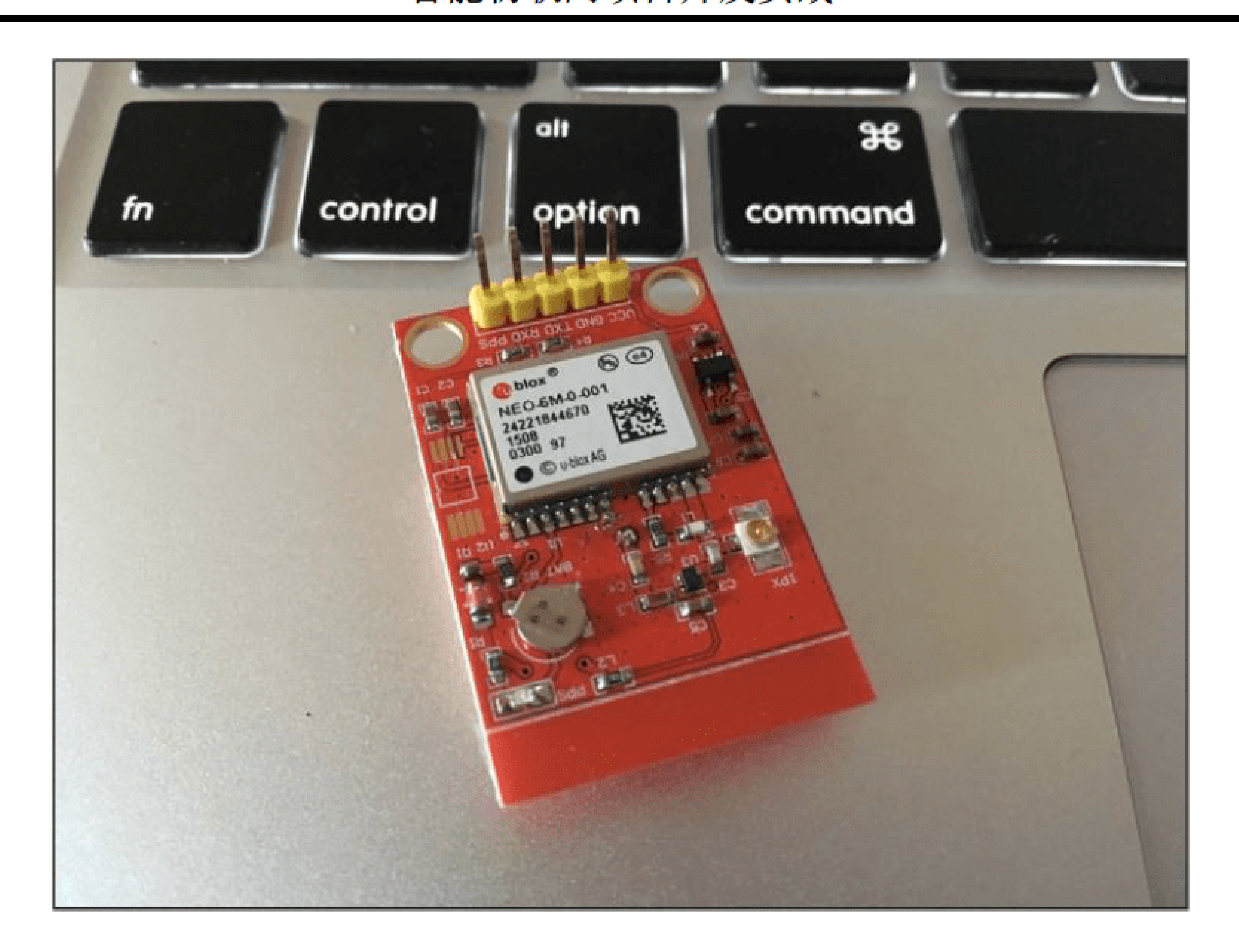


图 4-21

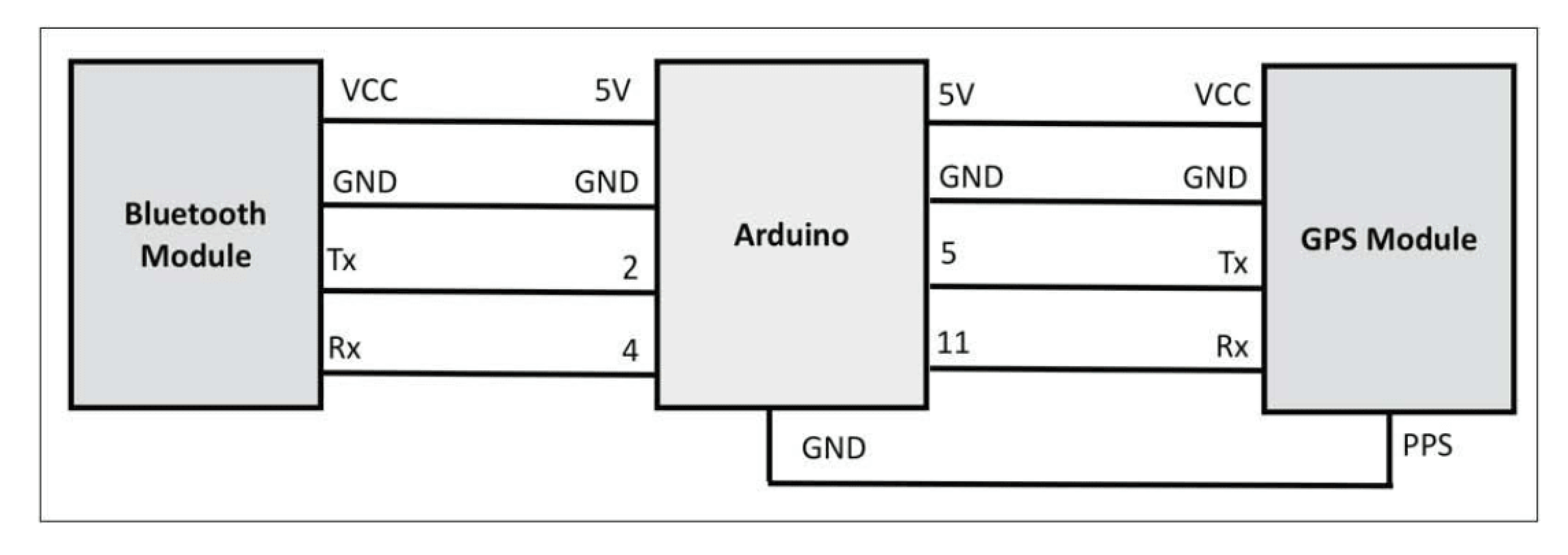


图 4-22

总结连线如下。

❑ 蓝牙模块，有如下连接：
  ➢ 蓝牙 VCC 连接到 Arduino 5V。
  ➢ 蓝牙 GND 连接到 Arduino GND。
  ➢ 蓝牙 Tx 连接到 Arduino pin 2。
  ➢ 蓝牙 Rx 连接到 Arduino pin 4。

❑ GPS 模块，有如下连接：
  ➢ GPS VCC 连接到 Arduino 5V。
  ➢ 蓝牙 GND 连接到 Arduino GND。

- 蓝牙 Tx 连接到 Arduino pin 5。
- 蓝牙 Rx 连接到 Arduino pin 11。
- 蓝牙 PPS 连接到 Arduino GND。

下一步是在 Arduino 上编写程序。打开 Arduino IDE，编写如下 Sketch 代码：

```
#include <SoftwareSerial.h>
#include <TinyGPS.h>

//D2   >>>  Rx, D4   >>>Tx
SoftwareSerialbluetooth(2, 4); //RX, TX

//D5   >>>  Rx, D11   >>>Tx
SoftwareSerialgps(5, 11);  //RX, TX
charval;
TinyGPSgps_mod;

void setup() {
Serial.begin(9600);
pinMode(13, OUTPUT);
bluetooth.begin(9600);
Serial.println("Bluetooth On..");
gps.begin(9600);
Serial.println("GPS On..");
}

void loop() {
boolnewData = false;
unsigned long chars;
unsigned short sentences, failed;

  //每 3s 读一次 GPS 位置
for (unsigned long start = millis(); millis() - start < 3000;) {
while (gps.available()){
char c = gps.read();
     //Serial.println(c);
if (gps mod.encode(c))
newData = true;
    }
  }

if (newData) {
float flat, flon;
```

```
unsigned long age;

digitalWrite(13, HIGH);
gps_mod.f_get_position(&flat, &flon, &age);
print_data("LAT=");
print_num_data(flat == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flat, 6);
print_data(" LON=");
print_num_data(flon == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flon, 6);
print_data(" SAT=");
print_num_data(gps_mod.satellites() == TinyGPS::GPS_INVALID_SATELLITES
? 0 : gps_mod.satellites());
print_data(" PREC=");
print_num_data(gps_mod.hdop()  ==  TinyGPS::GPS_INVALID_HDOP  ?  0  :
gps_mod.hdop());

break_line();
digitalWrite(13, LOW);
  }
}

voidprint_data(char msg[30]) {
Serial.print(msg);
bluetooth.print(msg);
}
voidprint_num_data(float msg,int n) {
Serial.print(msg, n);
bluetooth.print(msg, n);
}
voidprint_num_data(intmsg) {
Serial.print(msg);
bluetooth.print(msg);
}
voidbreak_line() {
Serial.println("");
bluetooth.println("");
}
```

把程序保存为 ch04_04。

然后可以编译并上传到 Arduino 板。把 Arduino 板添加到 Pololu Zumo robot。

在计算机上，应该建立 HC-06 和计算机的蓝牙连接，这在之前的章节中已经介绍过了。这个例子中，在笔者的苹果电脑上被识别为/dev/cu.HC-06-DevB。

现在编写一个 Python 程序来监听从 Arduino 板发来的 GPS 数据。

代码如下：

```
import serial
import sys

serial_hc06 = '/dev/cu.HC-06-DevB'
counter = 0

print('open ', serial_hc06)
hc06 = serial.Serial(serial_hc06, 9600)
print('read data from gps')
while True:
try:
        c = hc06.read(1)
if c != '':
sys.stdout.write(c)
sys.stdout.flush()

except (KeyboardInterrupt, SystemExit):
hc06.close()
raise

print('Exit')
```

把 serial_hc06 的值改为和蓝牙配对的 HC-06 的串口号码，然后保存程序为 ch04_05.py。运行如下命令进行测试：

```
$ python ch04_05.py
```

请确保 Arduino 板处于打开状态。

如果一切顺利，应该已经可以接收 Arduino 通过蓝牙发送来的 GPS 数据了。如图 4-23 所示是一个程序示例。

```
codes — python ch04_05.py — 80×20
agusk$ python ch04_05.py
('open ', '/dev/cu.HC-06-DevB')
read data from gps
LAT=-7.288106 LON=108.195365 SAT=10 PREC=87
LAT=-7.288107 LON=108.195365 SAT=11 PREC=74
LAT=-7.288111 LON=108.195373 SAT=11 PREC=74
LAT=-7.288119 LON=108.195365 SAT=11 PREC=74
LAT=-7.288127 LON=108.195365 SAT=11 PREC=74
LAT=-7.288125 LON=108.195358 SAT=11 PREC=74
LAT=-7.288122 LON=108.195358 SAT=11 PREC=74
LAT=-7.288122 LON=108.195358 SAT=11 PREC=74
```

图 4-23

如果 Arduino 是通过 USB 和计算机连接，可以使用串口监视工具看到消息。

如图 4-24 所示是串口监视器的一个示例输出。

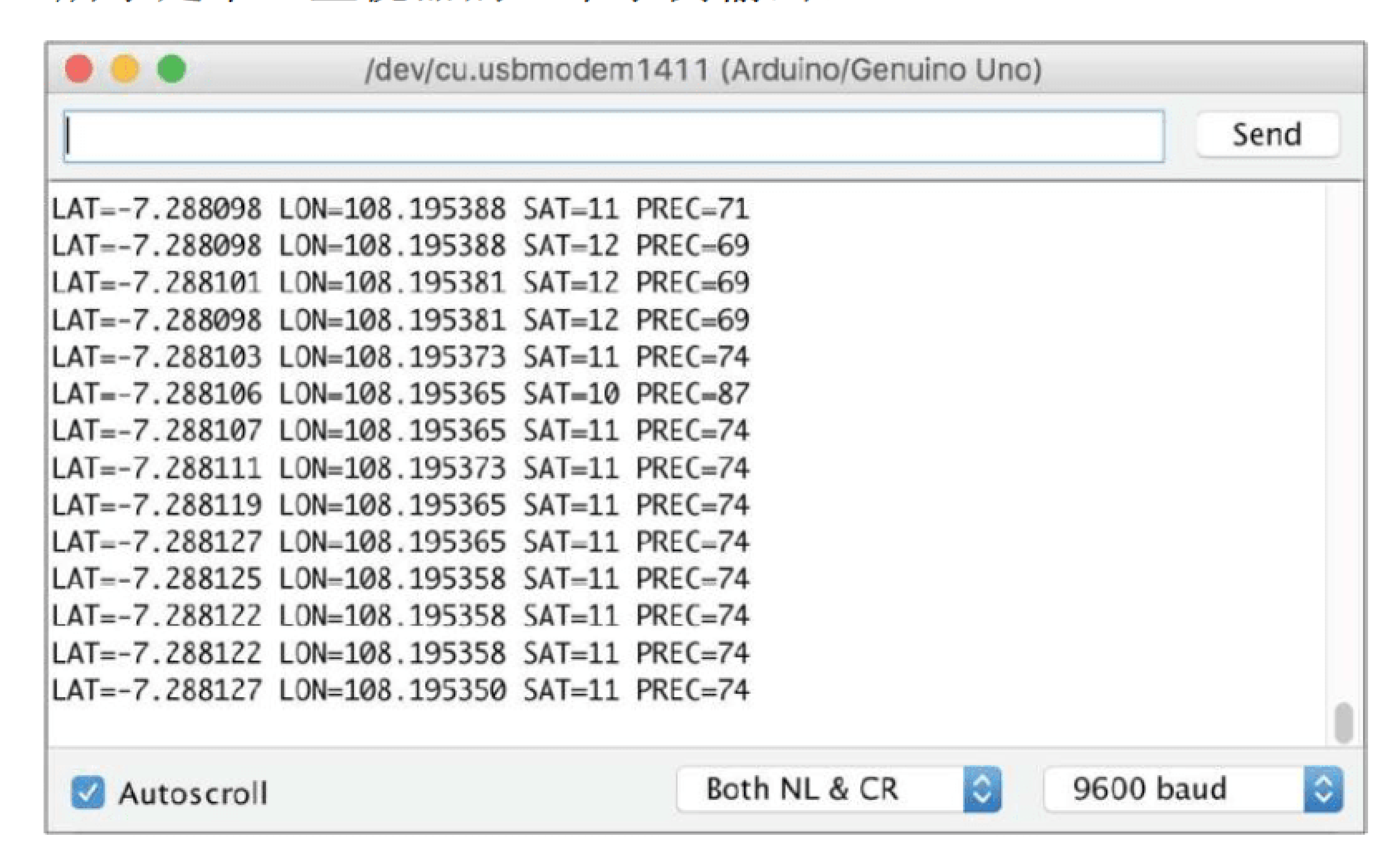


图 4-24

如何运行

运行方式几乎和之前的蓝牙例子一样，但是在这个例子里，给 Pololu Zumo robot 额外添加一个 GPS 模块。为了分析从卫星发来的 GPS 数据，这里使用了 TinyGPS 库。首先，在 Pololu Zumo robot 的 pins 5 和 11 定义 GPS 模块：

```
//D5   >>>  Rx, D11   >>>Tx
SoftwareSerialgps(5, 11);  //RX, TX
charval;
TinyGPSgps_mod;
```

在 setup()函数里初始化蓝牙和 GPS 模块，还包括串口对象。

```
void setup() {
Serial.begin(9600);
pinMode(13, OUTPUT);
bluetooth.begin(9600);
Serial.println("Bluetooth On..");
gps.begin(9600);
Serial.println("GPS On.."); }
```

从 GPS 模块上每 3s 获取一次数据，这个通过 loop()函数实现。

```
for (unsigned long start = millis(); millis() - start < 3000;) {
while (gps.available()){
```

```
char c = gps.read();
     //Serial.println(c);
     if (gps_mod.encode(c))  newData = true;
    }
  }
```

如果收到的是有效的 GPS 数据，就使用 TinyGPS 库进行解析，然后把得到的数据发送给蓝牙模块和 Arduino 串口 serial port。

```
if (newData) {
float flat, flon;
unsigned long age;

digitalWrite(13, HIGH);
gps_mod.f_get_position(&flat, &flon, &age);
print_data("LAT=");
print_num_data(flat == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flat, 6);
print_data(" LON=");
print_num_data(flon == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flon, 6);
print_data(" SAT=");
print_num_data(gps_mod.satellites() == TinyGPS::GPS_INVALID_SATELLITES
? 0 : gps_mod.satellites());
print_data(" PREC=");
print_num_data(gps_mod.hdop()  ==  TinyGPS::GPS_INVALID_HDOP  ?  0  :
  gps_mod.hdop());
break_line();
digitalWrite(13, LOW);   }
```

在计算机端，打开和 HC-06 匹配的串行端口。

```
serial_hc06 = '/dev/cu.HC-06-DevB'
counter = 0

print('open ', serial_hc06)
hc06 = serial.Serial(serial_hc06, 9600)
```

一旦端口打开，程序就会等待接受来自蓝牙的消息。只要收到信息，程序会立即打印到终端。

```
c = hc06.read(1)
if c != '':
sys.stdout.write(c)
sys.stdout.flush()
```

介绍地图引擎平台

为了可视化 GPS 数据，可以使用地图。把经度和纬度的值放在地图上，就可以准确地看到位置。

可以使用自己的地图来可视化 GPS 数据。本节将学习最通用的地图引擎平台 Google Maps API。这个库为在地图上可视化 GPS 数据提供了很多选择。可以访问 Google Maps API 的官方网站（https://developers.google.com/maps）寻找更多信息。

在此处的例子里，在 Python 程序里使用 Google Maps API。这里使用 Flask 库作为 Web 开发框架。可以在 http://flask.pocoo.org 中找到 Flask 库。

通过 pip 命令在计算机或者 Raspberry Pi 上安装 Flask 库：

```
$ pip install Flask
```

如果需要管理员权限安装，就在命令前面添加 sudo。在 Windows 平台，可以以管理员身份运行一个命令窗口。

下面来创建一个简单的 Flask 应用 gpsapp.py。输入如下命令：

```
$ mkdirgps_web
$ cdgps_web
$ nano gspapp.py
```

可以在 gpsapp.py 文件里输入如下代码：

```
from flask import Flask
from flask import render_template
from flask import jsonify

app = Flask(__name__)


@app.route('/hello')
defhello_world():
return 'Hello, World!'
```

保存程序，用如下命令运行程序：

```
$ export FLASK_APP=gpsapp.py
$ python -m flask run
```

程序默认会运行在 5000 端口，可以打开浏览器并访问 http://localhost:5000/hello，会在浏览器上看到“Hello, World!”。

如图 4-25 所示是在浏览器里看到的一个示例。

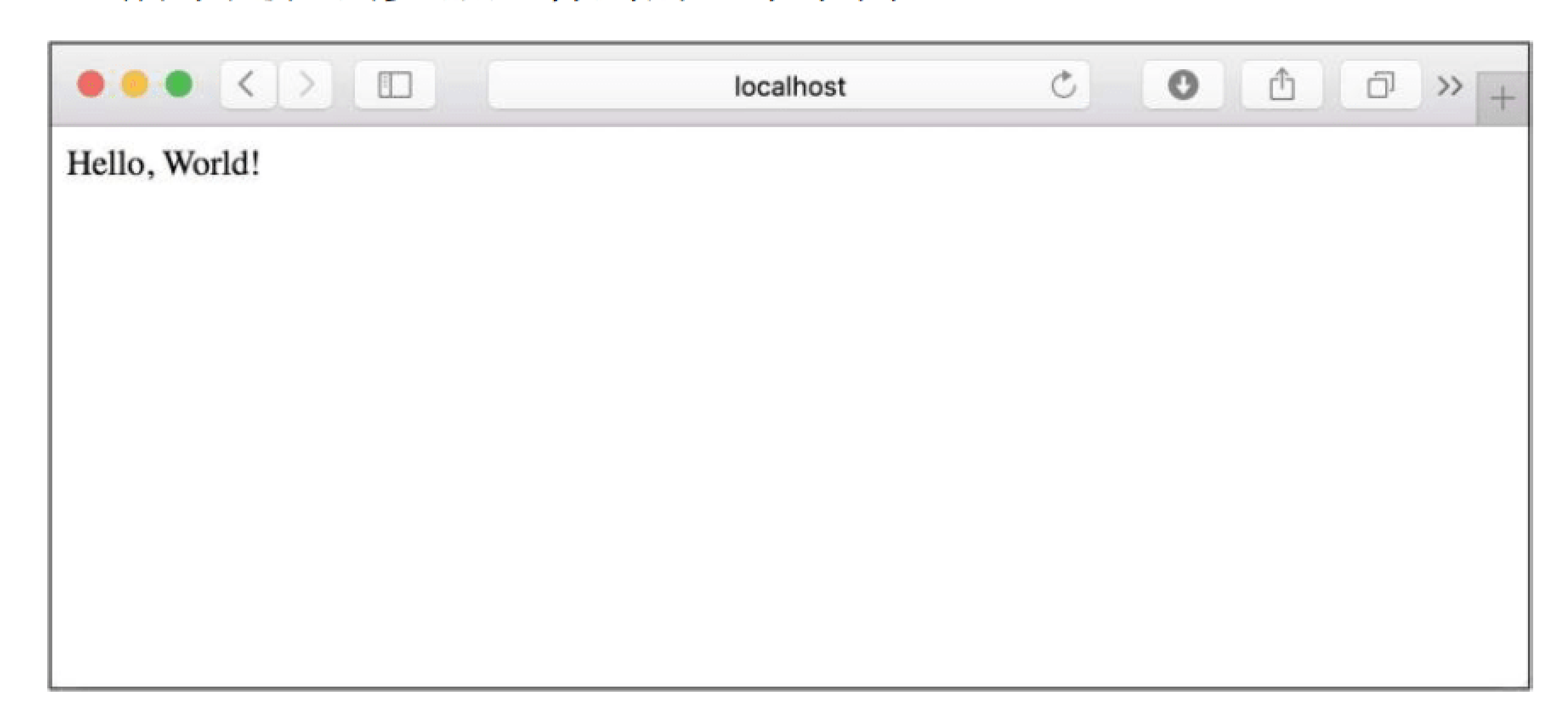


图 4-25

在程序的终端上，可以看到来自浏览器的请求消息。如图 4-26 所示是在终端的一个示例输出。

gps_web — Python -m flask run — 80×20

```
agusk$ python3 -m flask run
 * Serving Flask app "gpsapp"
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [06/Jul/2016 20:50:41] "GET /tracking HTTP/1.1" 200 -
127.0.0.1 - - [06/Jul/2016 20:50:42] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [06/Jul/2016 20:50:42] "GET /hello HTTP/1.1" 200 -
127.0.0.1 - - [06/Jul/2016 20:50:43] "GET /hello HTTP/1.1" 200 -
```

图 4-26

如何运行

在程序中，gpsapp.py 文件里定义了一个路由/hello 并且对请求返回“Hello World!”字符串：

```
@app.route('/hello')
defhello_world():
return 'Hello, World!'
```

继续在文件 gpsapp.py 里集成谷歌地图。

在程序文件夹里创建一个 template 文件夹，然后新建一个文件 index.html。在 index.html 文件里写进如下脚本：

```
<!doctype html>
<html lang="en">
<head>
<title>Google Maps Demo</title>
<meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
<scriptsrc="http://maps.google.com/maps/api/js?sensor=false"></script>
<script>
var map;
function initialize() {
varmyCenter = new google.maps.LatLng(52.524343, 13.412751);
map = new google.maps.Map(document.getElementById('map'), {
zoom: 5,
center: myCenter,
mapTypeId: google.maps.MapTypeId.ROADMAP
       });
var marker=new google.maps.Marker({
position:myCenter
       });
marker.setMap(map);
   }
</script>
<style>
body {font-family: sans-serif}
       #map {width: 640px; height: 480px}
</style>
</head>
<body onload='initialize()'>
<div id=map></div>
</body>
</html>
```

脚本会通过 JavaScript 加载谷歌地图 API。接着创建一个 google.maps.Map 的对象，并传入一个经纬度坐标，如 52.524343、13.412751。当然可以随意更改这个数值，然后利用 google.maps.Marker 对象在经纬度坐标的位置添加一个标记。

现在修改 gpsapp.py 文件添加一个额外的路由。如下是 gpsapp.py 文件的完整版。

```
from flask import Flask
from flask import render_template
from flask import jsonify

app = Flask(__name__)


@app.route('/hello')
defhello_world():
return 'Hello, World!'


@app.route('/')
def index(name=None):
returnrender_template('index.html', name=name)
```

现在可以在终端里运行 gpsapp.py 文件了。接着打开浏览器并访问 http://localhost:5000/。请确保计算机连接着网络，因为程序 index.html 需要请求谷歌地图 API。

如图 4-27 所示是在程序在浏览器里的示例输出。

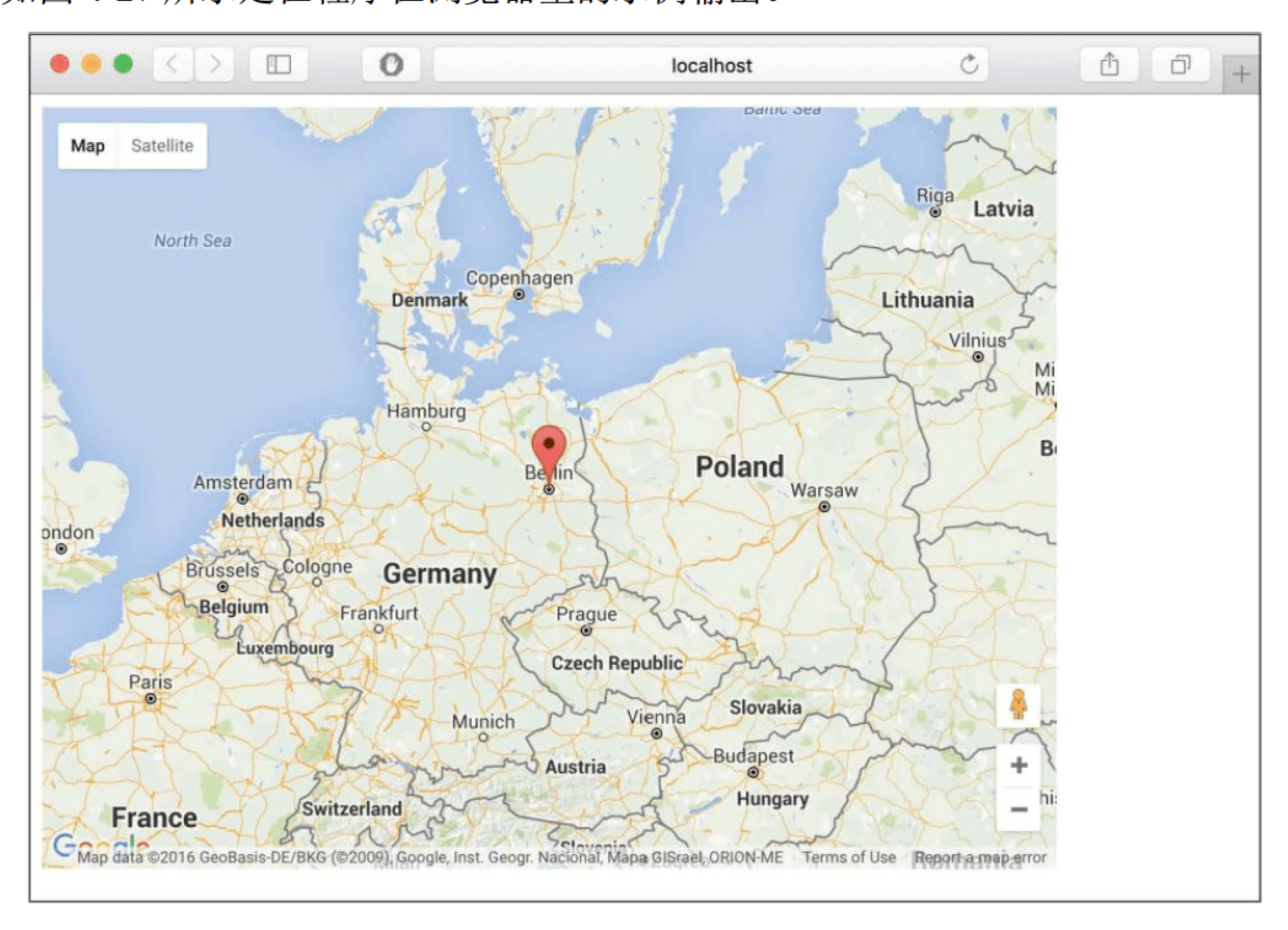


图 4-27

制作基于 GPS 的小车

在之前的章节里，我们学会了如何在 Python 程序里使用 Flask 库访问谷歌地图。现在将结合之前在 Pololu Zumo robot 上读取 GPS 数据的知识。

在程序从 Arduino 上读取了 GPS 数据，可以发送 GPS 数据到 Web 服务器。

首先，修改 gpsapp.py 文件来读取 GPS 数据。可以直接从 GPS 模块读取或者在计算机上通过中间件软件读取。

在 get_gps_data()函数里定一个路由。这个函数可以起到让值"hardcoded"的效果。会从 GPS 模块得到 lat_val 和 long_val，get_gps_data()将返回一个 JSON 格式的值。这使得 gspapp.py 达到了 RESTful 服务的效果。

如下是 gpsapp.py 程序的完整代码：

```
from flask import Flask
from flask import render template
from flask import jsonify

app = Flask(__name__)


@app.route('/hello')
defhello_world():
return 'Hello, World!'


@app.route('/')
def index(name=None):
returnrender template('index.html', name=name)


@app.route('/tracking')
def tracking(name=None):
returnrender_template('tracking.html', name=name)


@app.route('/gps', methods=['GET'])
defget_gps_data():

    # 通过无线模块例如 GSM、WiFi、Bluetooth 从机器人获取数据
```

```
lat_val = 52.524343
long_val = 13.412751

returnjsonify(lat=lat_val, long=long_val)
```

保存程序。

下一步将创建一个 HTML 文件。新建一个文件 tracking.html，并把它放到 templates 文件夹下。

和 index.html 里不同的是，tracking.html 里经纬度的值来自服务器。我们将利用 jQuery 从服务器获取数据。通过调用 setInterval()函数每 5s 访问一次 RESTful 服务：

```
<!doctype html>
<html lang="en">
<head>
<title>Robot Tracking Demo</title>
<meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.
min.js"> </script>
<scriptsrc="http://maps.google.com/maps/api/js?sensor=false"></script>
<style>
body {font-family: sans-serif}
       #map {width: 640px; height: 480px}
</style>
</head>
<body>
<div id=map></div>

<script>
var map;
varisLoading = false;

functionload_maps(lat,long) {
varmyCenter = new google.maps.LatLng(lat, long);
map = new google.maps.Map(document.getElementById('map'), {
zoom: 5,
center: myCenter,
mapTypeId: google.maps.MapTypeId.ROADMAP
          });
var marker=new google.maps.Marker({
position:myCenter
          });
```

```
marker.setMap(map);
isLoading = false;
        }


setInterval(function(){
if(!isLoading) {
isLoading = true;
                $.getJSON("/gps", function(result){
load_maps(result.lat, result.long);
                });

            }
        }, 5000);
</script>
</body>
</html>
```

保存程序。

现在可以运行程序并访问 http://localhost:5000/tracking，应该可以看到一个有注释和标注的完整地图。页面每 5s 刷新一次。

你已经知道如何利用浏览器追踪机器车。那么可以从一个网站来控制机器车吗？可以作为练习试一试。

制作自动机器车

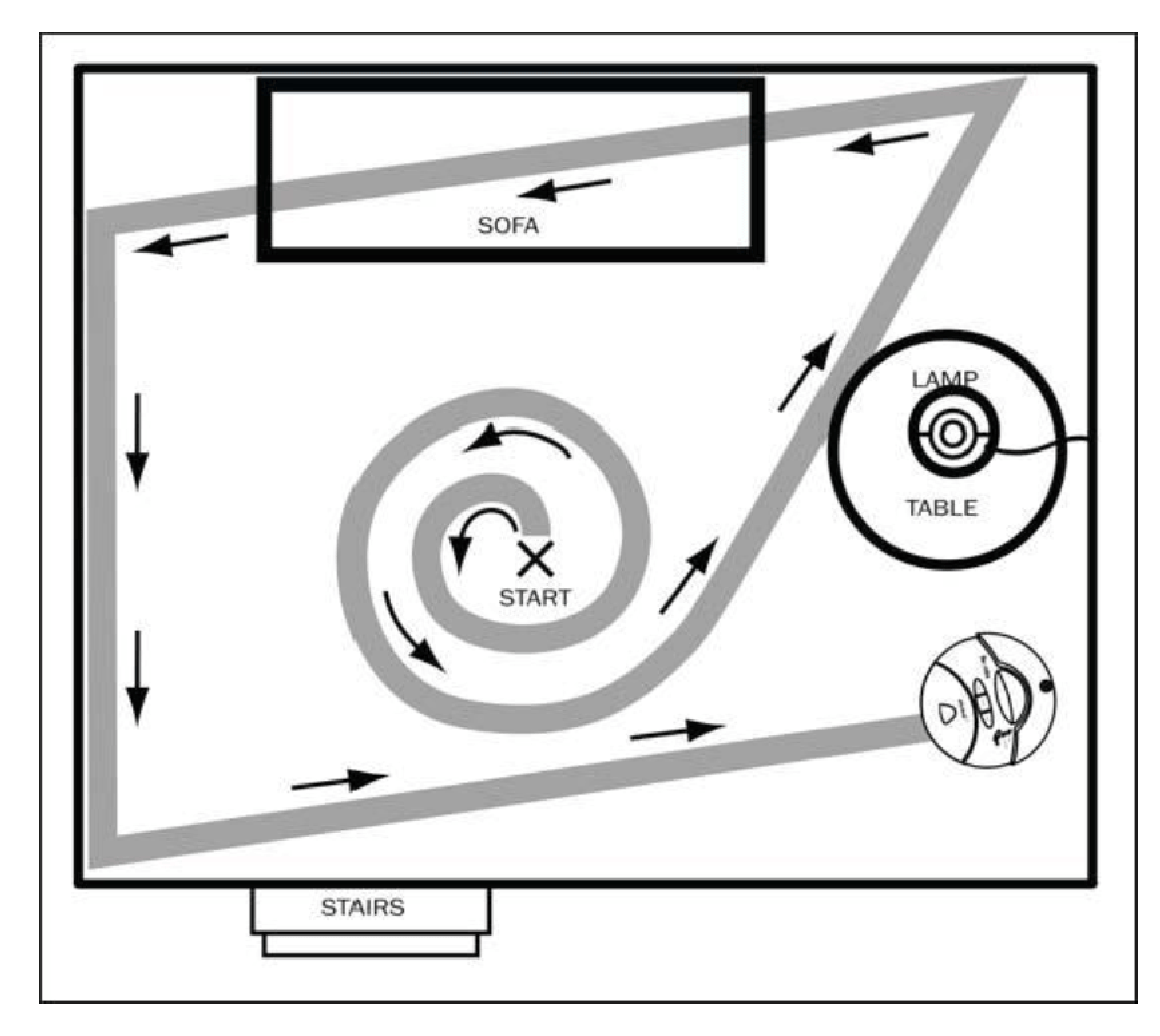
所有制作机器车的材料都已经具备了。现在可以基于自己的需求来制作。你可以制作一个自动的满足特定任务的移动机器，举个例子，一个真空吸尘机器车。可以用 Ultrasonic 模块来检测障碍物。

在笔者看来，制作一个真空吸尘机器人最大的困难是设计清洁路径算法——如何让机器车访问到所有的路。由于机器车没有地图，故可以用一个 microSD 卡存储走过的路，可以用中间件的方式引导机器车走一条路。

看一看 iRobot http://www.irobot.com/For-the-Home/Vacuuming/Roomba.aspx 网站上的一个 Roomba 的例子。如图 4-28 所示是 Roomba robot 用的清洁路径算法。

阅读本章“引用”部分的论文（4，5，6）来寻找制作真空清洁机器车的灵感。

通过集成传感器和驱动器，可以拓展制作机器车的想象力。



来源：https://www.irobot.com/

图 4-28

总　　结

在本章中，学到了一些制作机器车的基本技术，也探索了一些机器人制作项目的平台，我们选择了其中一个平台 Pololu Zumo robot for Arduino 作为练习，还在机器车上集成导航模块。在最后一个主题中，还使用地图引擎谷歌地图 API 来追踪机器车。

在第 5 章，将学习如何在物联网项目里集成语音技术。

引　　用

下面是一些推荐的论文、数据和网站，读者可以了解更多关于本章的内容。

1. Meystel, A.: *Intelligent Control: A Sketch of the Theory, 1. Intelligent and Robotic Systems*, Vol. 2, 1989, pp. 97-107.

2. *Autonomous Robotic Systems. Lecture Notes in Control and Information Sciences*, Springer, 1998.

3. *Advances in Intelligent Autonomous System. International Series on MICROPROCESSOR-BASED AND INTELLIGENT SYSTEMS ENGINEERING*, Vol. 18, Springer Science + Business Media Dordrecht, 1999.

4. K. M. Hasan, Abdullah-Al-Nahid and K. J. Reza, *Path planning algorithm development for autonomous vacuum cleaner robots*, Informatics, Electronics & Vision (ICIEV), 2014 International Conference on, Dhaka, 2014, pp. 1–6.

5. Sewan Kim, *Autonomous cleaning robot: Roboking system integration and overview*, Robotics and Automation, 2004. Proceedings, ICRA '04.2004 IEEE International Conference on, 2004, pp. 4437-41 Vol. 5.

6. Ryo Kurazume, Shigeo Hirose, *Development of a Cleaning Robot System with Cooperative Positioning System* in Autonomous Robots (2000) Volume 9, Issue: 3, Springer, pp. 237-46.

7. Flask, http://flask.pocoo.org.

8. Google Maps API, https://developers.google.com/maps.

第 5 章　在物联网项目中添加语音技术

本章将探索如何让物联网项目“说”点什么，同时在项目中研究各种语音和声音模块的使用。

分为如下几个主题：

- 语音技术介绍
- 声音传感器和驱动器介绍
- 语音技术的模式识别介绍
- 回顾语音和声音模块
- 为物联网项目增加语音控制
- 让物联网项目说话
- 让 Raspberry Pi 说话

语音技术介绍

语音是人们交流的主要途径之一。语音技术的核心是语音识别。机器，如计算机，可以识别出人们说了什么，甚至可以识别每个人说话的不同方式从而区分不同人的语音。

语音技术包含从语音到文本和从文本到语音两个方面。研究者们已经为一些语言构建了语音模块，如英语、德语、汉语和法语。

如图 5-1 所示就是语音技术的示意图。

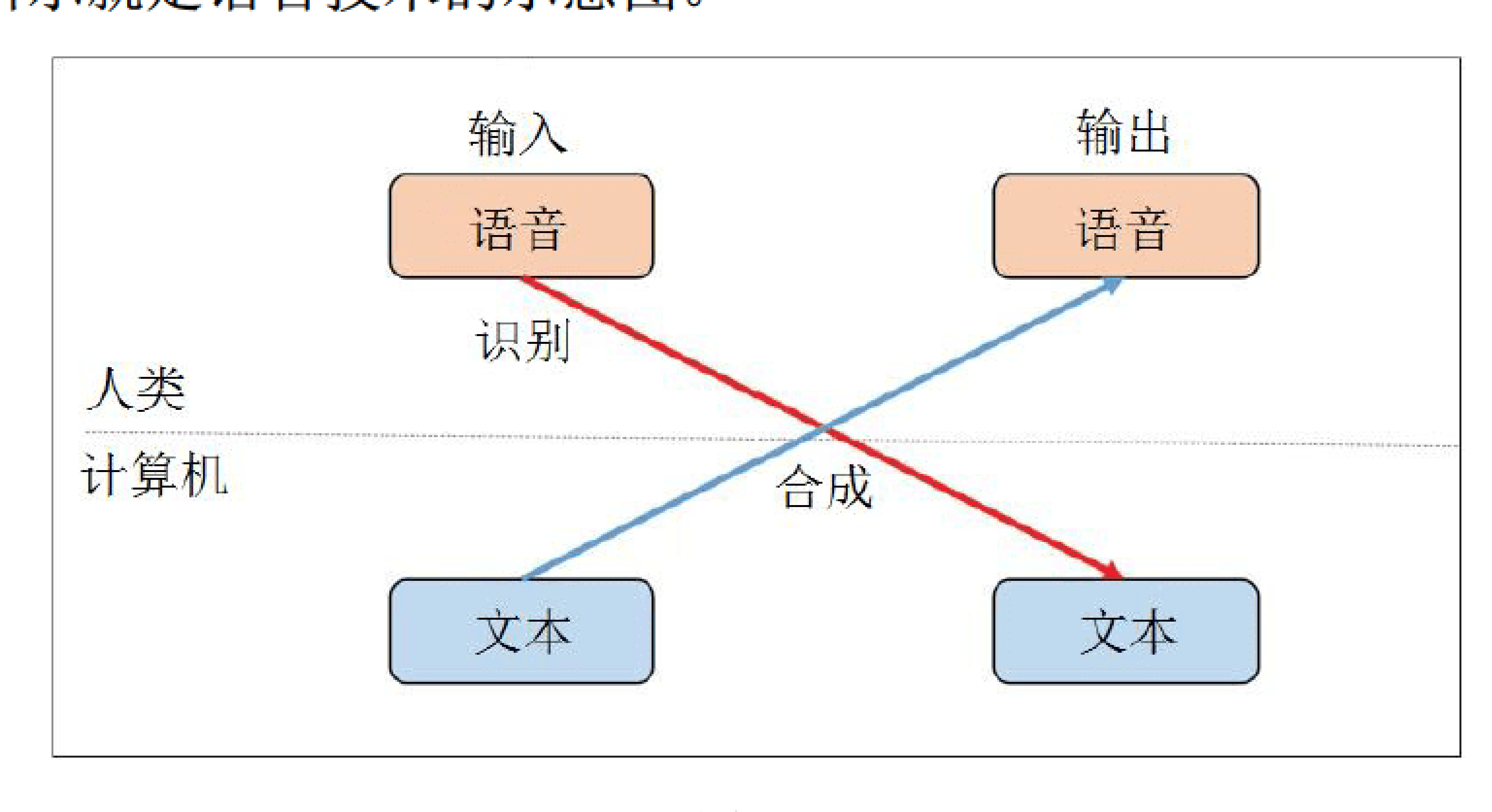


图 5-1

为了把语音转换为文本，需要使用语音识别技术，而对于从文本生成语音，则需要语音合成技术。本书不会涵盖语音识别和语音合成领域繁杂的数学公式和统计方法，但强烈建议读者阅读相关资料。

在本章中，将学习如何在物联网平台的环境下处理声音和语音。

声音传感器和驱动器介绍

声音可以来自人类、动物、汽车等。为了处理声音数据，首先把声音来源从物理形式转化为数字形式。这个可以用传感设备捕捉声音来源来实现，最简单的声音传感器就是麦克风，麦克风可以记录任意的声音来源。

把麦克风模块连接到物联网板子上，如 Arduino 和 Raspberry Pi。其中的模块之一，如 Electret Microphone Breakout（网址为 https://www.sparkfun.com/products/12758），这个模块有 3 个 pin 出口：AUD、GND 和 VCC，如图 5-2 所示。



图 5-2

更进一步，可以用一个驱动器生成声音，一个简单的声音驱动器是蜂鸣器，可以在有限的频率内生成简单的声音。可以通过模拟输出或者 PWM pin 发送数字 pin 来生成声音。一些厂商也为蜂鸣器提供 breakout 模块。如图 5-3 所示是一个蜂鸣器驱动器的形状。

蜂鸣器通常是一个被动驱动器，如果想要一个主动的声音驱动器，可以使用扬声器。这个组件在当地或者网上商店都非常容易买到。笔者在 Sparkfun 上买了一个，网址为 https://www.sparkfun.com/products/11089，如图 5-4 所示。

为了获取使用声音传感器/驱动器的经验，下面给出一个捕捉声音来源强度的例子。

在这个例子里，将展示如何使用声音传感器 Electret 麦克风检测声音强度的级别。这个声音的来源可以包括唱歌、鼓掌、敲门或者其他能被传感器捕捉到的足够大的声音。

传感器的输出是模拟值，这样 MCU 可以通过一个模拟到数字转换器进行转换。



图 5-3



图 5-4

下面是这个例子中要用到的一些外部设备：

- Arduino 板。
- Resistor 330 Ohm。
- Electret Microphone Breakout，网址为 https://www.sparkfun.com/products/12758。
- 10 个 Segment LED Bar Graph -红色，网址为 https://www.sparkfun.com/products/9935。可以使用任意颜色。

可以使用 Adafruit Electret Microphone Breakout 添加到 Arduino 板子。参见 https://www.adafruit.com/product/1063。

为了制作例子，需要把如下部件连线：

- 连接 Electret Microphone AUD pin 到 Arduino A0 pin。
- 连接 Electret Microphone GND pin 到 Arduino GND pin。
- 连接 Electret Microphone VCC pin 到 Arduino 3.3V pin。
- 分别连接 10 Segment LED Bar Graph pins 到 Arduino digital pins：3, 4, 5, 6, 7, 8, 9, 10, 11, 12。

最终的接线效果如图 5-5 所示。

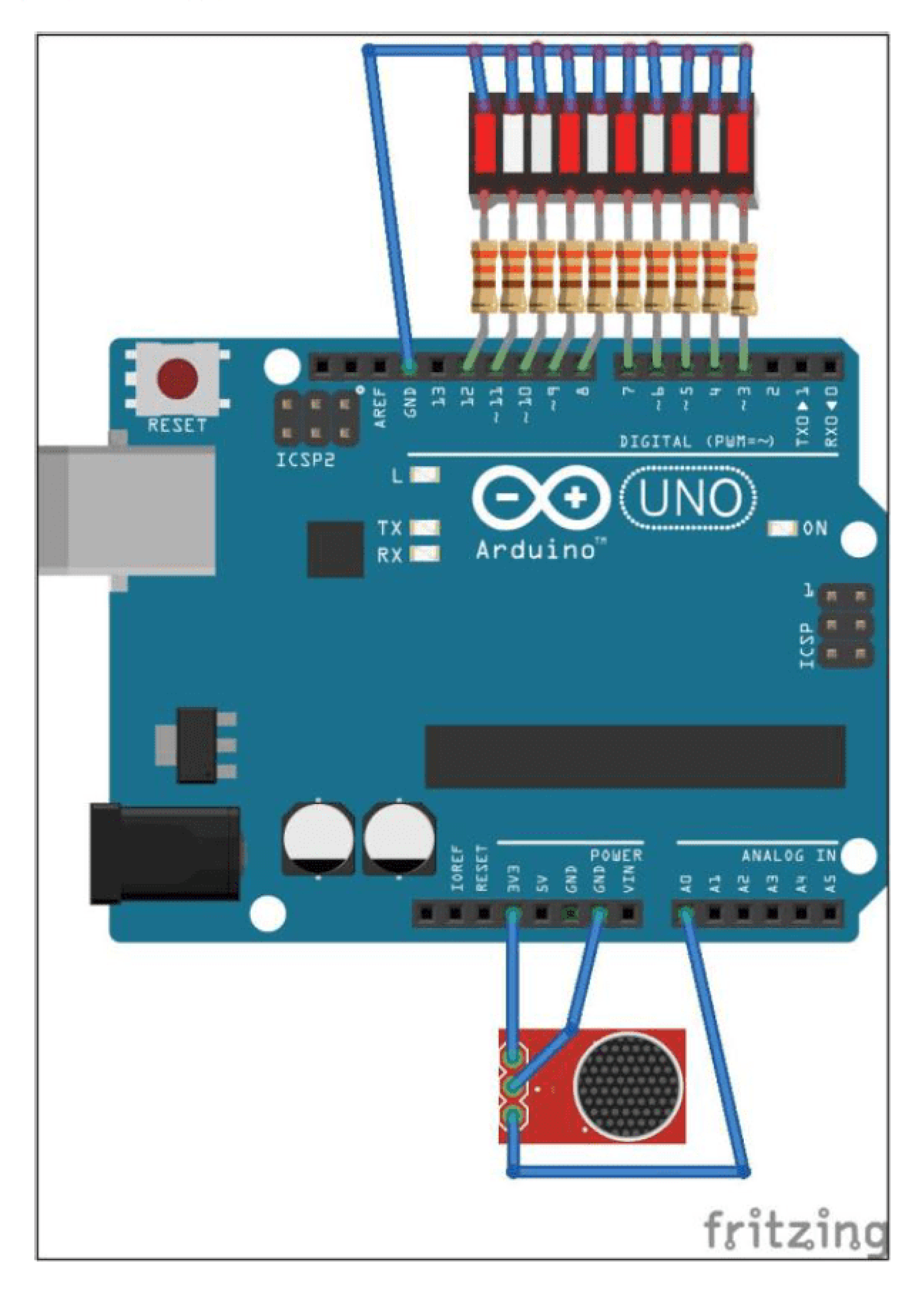


图 5-5

这 10 个独立的 Led Bar Graph 模块用来代表声音强度的级别。在 Arduino 中可以用 analogRead()函数从外部传感器读取模拟输入。analogRead()返回值为 0～1023。

累计的输出电压是 3.3V，因为把 Electret Microphone Breakout 连接到 3.3V 的 VCC。这样，可以为每个 led bar 设置 3.3/10 = 0.33 的电压。第一个 led bar 被连接到 Arduino 数组 pin 3。

现在可以实现 sketch 程序来读取声音的强度，然后将其转换到 10 Segment Led Bar Graph。

为了获取声音强度，应从模拟输入 pin 读取声音输入。每次读取持续一段时间，叫作采样窗口时间，如 250ms。在这段时间里，得到模拟输入的峰值（最大值），这个峰值就代表声音强度的值。

开始实现程序。打开 Arduino IDE，编写如下 Sketch 程序。

```
//用时间代表采样窗口大小（250mS，相当于 4Hz）
const int sampleWindow = 250;
unsigned int sound;
int led = 13;

void setup()
{
   Serial.begin(9600);
   pinMode(led, OUTPUT);

   pinMode(3, OUTPUT);
   pinMode(4, OUTPUT);
   pinMode(5, OUTPUT);
   pinMode(6, OUTPUT);
   pinMode(7, OUTPUT);
   pinMode(8, OUTPUT);
   pinMode(9, OUTPUT);
   pinMode(10, OUTPUT);
   pinMode(11, OUTPUT);
   pinMode(12, OUTPUT);
}

void loop()
{
  unsigned long start= millis();
  unsigned int peakToPeak = 0;

  unsigned int signalMax = 0;
  unsigned int signalMin = 1024;

  //持续采集 250s
  while (millis() - start < sampleWindow)
  {
    sound = analogRead(0);
```

```
    if (sound < 1024)
    {
      if (sound > signalMax)
      {
        signalMax = sound;
      }
      else if (sound < signalMin)
      {
        signalMin = sound;
      }
    }
  }
  peakToPeak = signalMax - signalMin;
  double volts = (peakToPeak * 3.3) / 1024;

  Serial.println(volts);
  display_bar_led(volts);
}

void display_bar_led(double volts)
{
  display_bar_led_off();

  int index = round(volts/0.33);
  switch(index){
    case 1:
      digitalWrite(3, HIGH);
      break;
    case 2:
      digitalWrite(3, HIGH);
      digitalWrite(3, HIGH);
      break;
    case 3:
      digitalWrite(3, HIGH);
      digitalWrite(4, HIGH);
      digitalWrite(5, HIGH);
      break;
    case 4:
      digitalWrite(3, HIGH);
      digitalWrite(4, HIGH);
      digitalWrite(5, HIGH);
      digitalWrite(6, HIGH);
```

```
    break;
  case 5:
    digitalWrite(3, HIGH);
    digitalWrite(4, HIGH);
    digitalWrite(5, HIGH);
    digitalWrite(6, HIGH);
    digitalWrite(7, HIGH);
    break;
  case 6:
    digitalWrite(3, HIGH);
    digitalWrite(4, HIGH);
    digitalWrite(5, HIGH);
    digitalWrite(6, HIGH);
    digitalWrite(7, HIGH);
    digitalWrite(8, HIGH);
    break;
  case 7:
    digitalWrite(3, HIGH);
    digitalWrite(4, HIGH);
    digitalWrite(5, HIGH);
    digitalWrite(6, HIGH);
    digitalWrite(7, HIGH);
    digitalWrite(8, HIGH);
    digitalWrite(9, HIGH);
    break;
  case 8:
    digitalWrite(3, HIGH);
    digitalWrite(4, HIGH);
    digitalWrite(5, HIGH);
    digitalWrite(6, HIGH);
    digitalWrite(7, HIGH);
    digitalWrite(8, HIGH);
    digitalWrite(9, HIGH);
    digitalWrite(10, HIGH);
    break;
  case 9:
    digitalWrite(3, HIGH);
    digitalWrite(4, HIGH);
    digitalWrite(5, HIGH);
    digitalWrite(6, HIGH);
    digitalWrite(7, HIGH);
    digitalWrite(8, HIGH);
```

```
      digitalWrite(9, HIGH);
      digitalWrite(10, HIGH);
      digitalWrite(11, HIGH);
      break;
    case 10:
      digitalWrite(3, HIGH);
      digitalWrite(4, HIGH);
      digitalWrite(5, HIGH);
      digitalWrite(6, HIGH);
      digitalWrite(7, HIGH);
      digitalWrite(8, HIGH);
      digitalWrite(9, HIGH);
      digitalWrite(10, HIGH);
      digitalWrite(11, HIGH);
      digitalWrite(12, HIGH);
      break;
  }

}

void display_bar_led_off()
{
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
  digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  digitalWrite(8, LOW);
  digitalWrite(9, LOW);
  digitalWrite(10, LOW);
  digitalWrite(11, LOW);
  digitalWrite(12, LOW);
}
```

把 Sketch 程序保存为 ch05_01。

编译并部署到 Arduino 板。

部署完程序，可以打开 Serial Plotter 工具，此工具可以在 Arduino menu Tools -| Serial Plotter 找到。在工具上设置波特率为 9600 波特。

在声音传感器设备周围制造声音。可以从 Serial Plotter tool 上看到值的变化。如图 5-6 所示是 Serial Plotter 的一个示例输出。

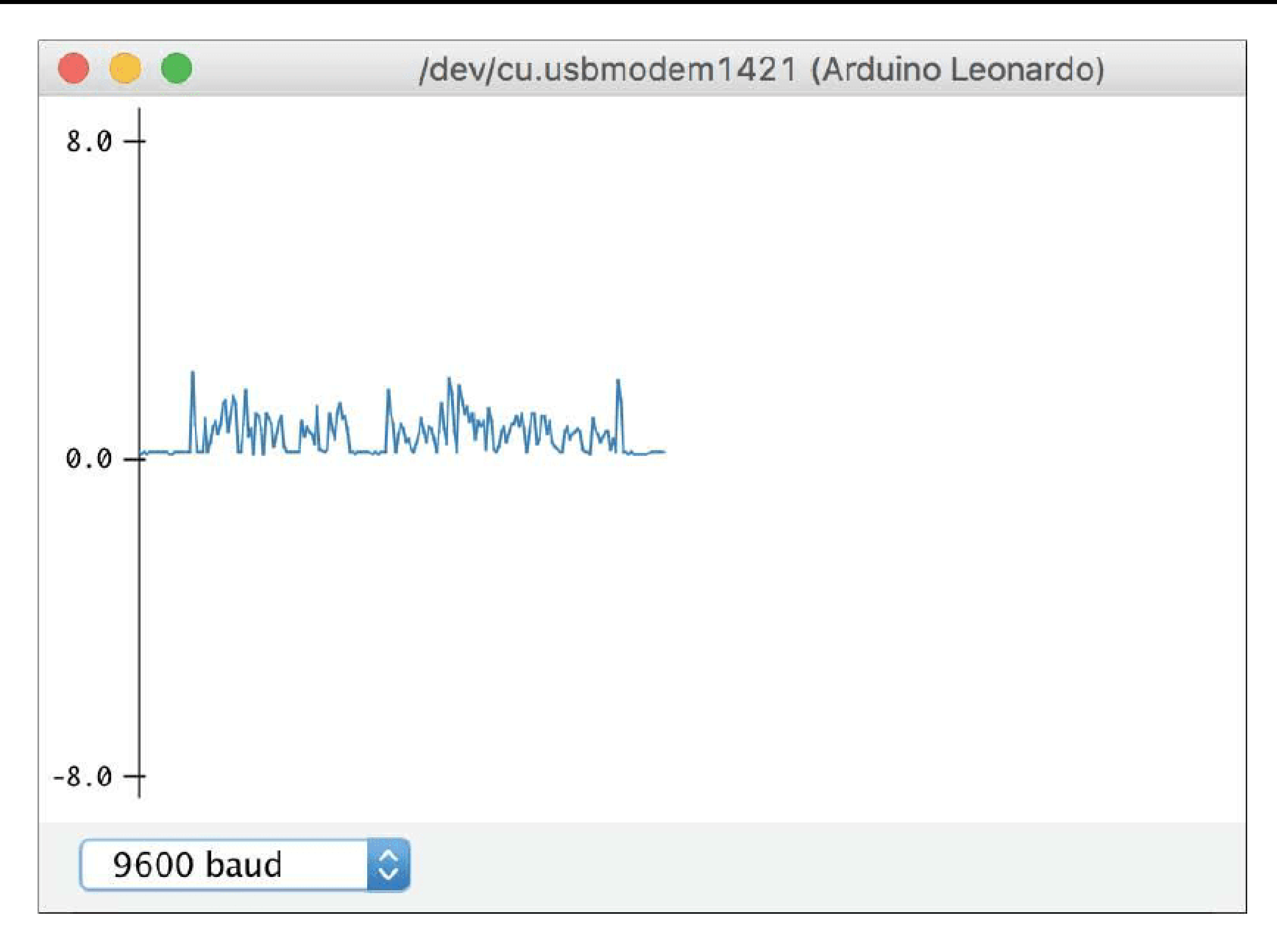


图 5-6

如何工作？

获取声音强度的思路很简单。要在声音的峰值里获取一个值。首先定义一个采样宽度，如 250ms 代表 4Hz。

```
//用时间代表采样窗口大小（250ms=4Hz）
const int sampleWindow = 250;
unsigned int sound;

int led = 13;
```

在 setup()函数里，初始化串行端口和 10 Segment Led Bar Graph。

```
void setup()
{
   Serial.begin(9600);
   pinMode(led, OUTPUT);

   pinMode(3, OUTPUT);
   pinMode(4, OUTPUT);
   pinMode(5, OUTPUT);
   pinMode(6, OUTPUT);
   pinMode(7, OUTPUT);
   pinMode(8, OUTPUT);
   pinMode(9, OUTPUT);
```

```
 pinMode(10, OUTPUT);
 pinMode(11, OUTPUT);
 pinMode(12, OUTPUT);

}
```

在 loop()函数里，在采样窗口中计算声音的强度。获取 peak-to-peak 值之后将其转换为电压形式。

```
unsigned long start= millis();
unsigned int peakToPeak = 0;

unsigned int signalMax = 0;
unsigned int signalMin = 1024;

//每 250ms 收集一次数据
while (millis() - start < sampleWindow)
{
  sound = analogRead(0);
  if (sound < 1024)
  {
    if (sound > signalMax)
    {
     signalMax = sound;
    }
    else if (sound < signalMin)
    {
      signalMin = sound;
    }
 }
}
peakToPeak = signalMax - signalMin;

double volts = (peakToPeak * 3.3) / 1024;
```

然后，通过调用 display_bar_led()在串口和 10 Segment Led 里以电压形式展示声音强度。

```
Serial.println(volts);

display_bar_led(volts);
```

在 display_bar_led()函数里，通过调用 display_bar_led_off()关闭 10 Segment Led Bar Graph，其内部实现是通过 digitalWrite()发送 LOW 给所有的 LEDs。之后从电压里计算一个范围值，这个值将被转换到 LEDs.

```
display bar led off();
int index = round(volts/0.33);
```

语音技术的模式识别介绍

模式识别是机器学习的一个主要方向，也是语音识别的基础。通常，按照图 5-7 所示构建语音识别系统。

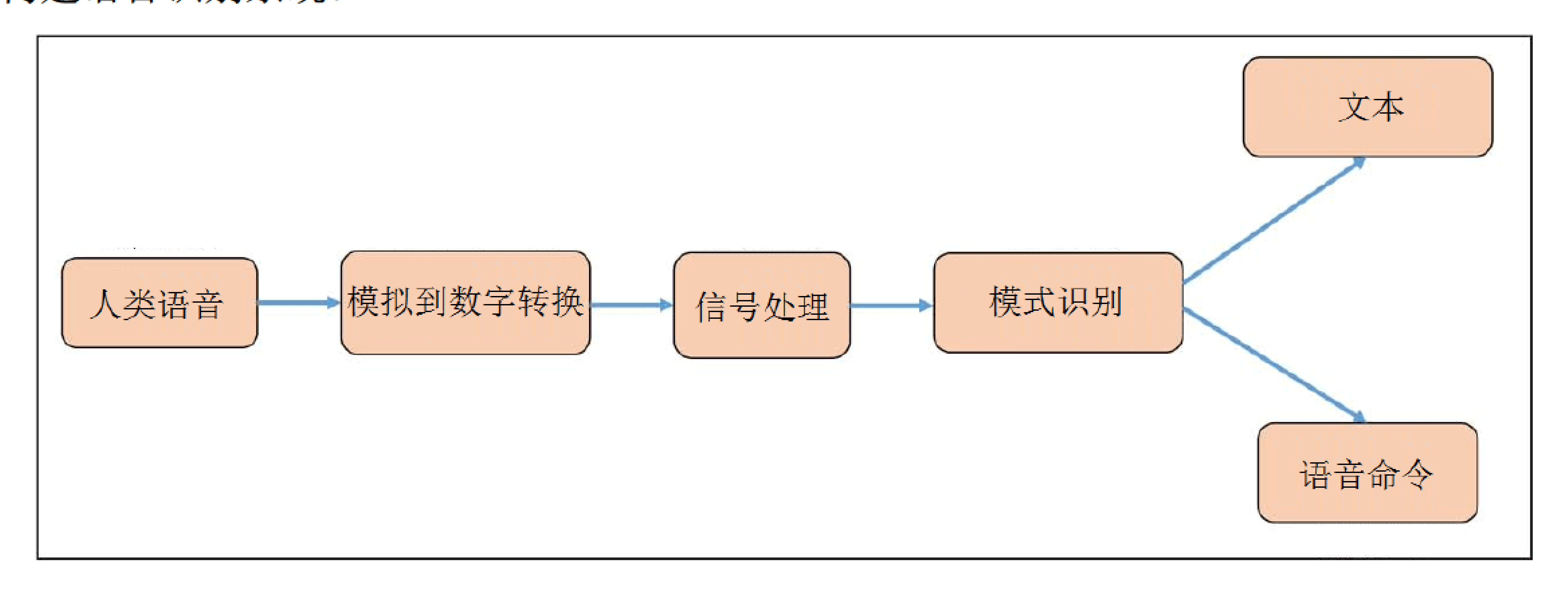


图 5-7

人类的语音会被转为数字形式，称之为离散数据。通常会使用一些技术，如去噪等对数据进行预处理。

现在利用模式识别进行语音识别。研究者们做了一些方法，如隐含马尔科夫模型来识别声音和词的关系。语音数字数据的特征抽取是模式识别的一部分。抽取的特征将会作为模式识别的输入。

模式识别技术被用在我们的物联网项目中的 Speech-to-Text 和 Speech 命令里。

介绍语音和声音模块

在本章中，将回顾各种可以被集成到 MCU 板的语音和声音模块。有许许多多和语音声音处理相关的模块，每个模块都有适合项目的特性。

其中一个模块是 VeeaR 的 EasyVR 3 & EasyVR Shield 3。更多相关信息可以查看 http://www.veear.eu/introducing-easyvr-3-easyvr-shield-3/。目前已经支持英语（US）、意大利语、德语、法语、西班牙语和日语。

如图 5-8 所示是 EasyVR 3 模块的一个示例。

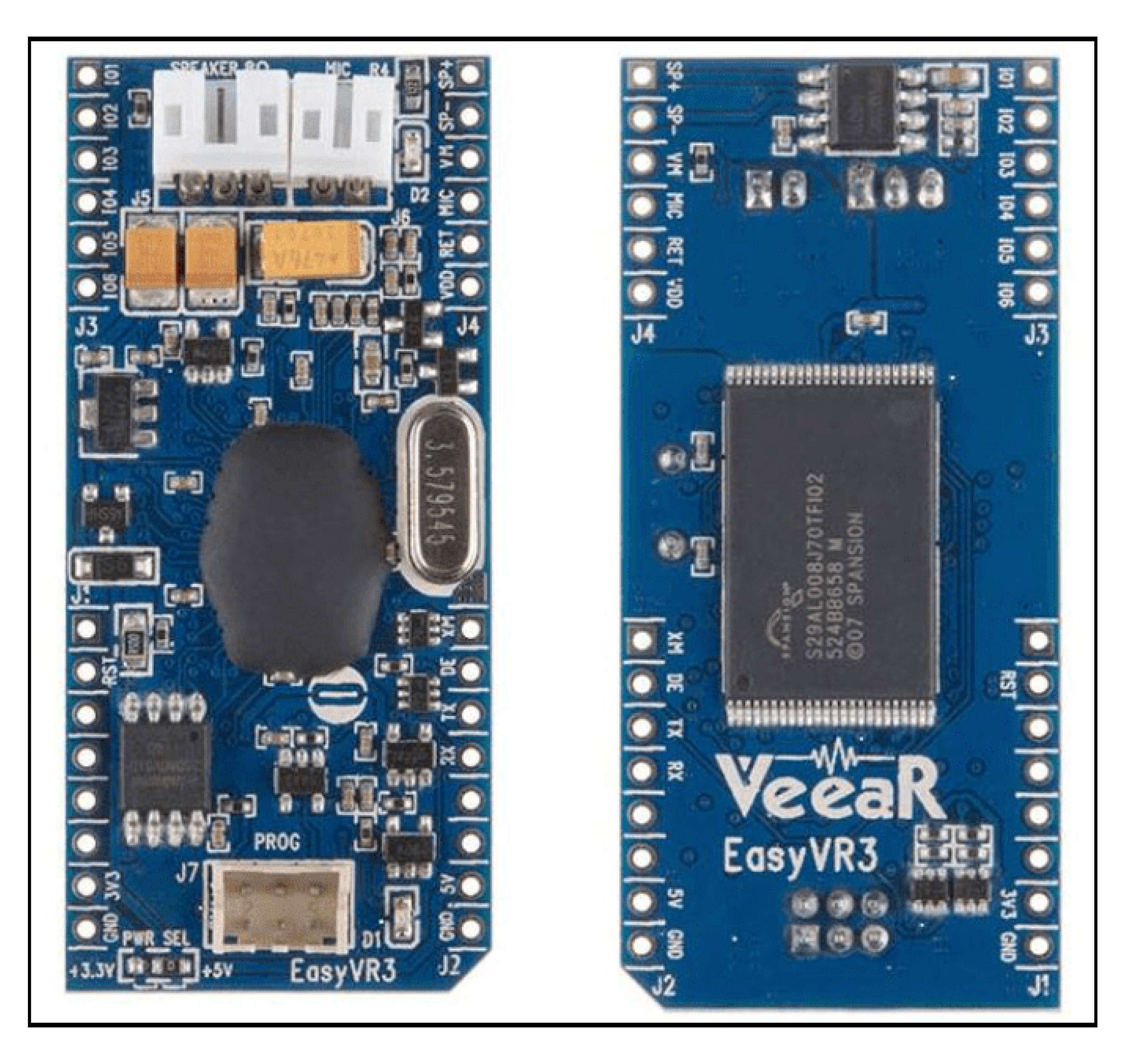


图 5-8

EasyVR 3 板也适合作为 Arduino shield。如果买了一个 EasyVR Shield 3，会同时获得一个 EasyVR 板和 Arduino shield。如图 5-9 所示是 EasyVR Shield 3 的一个示例。

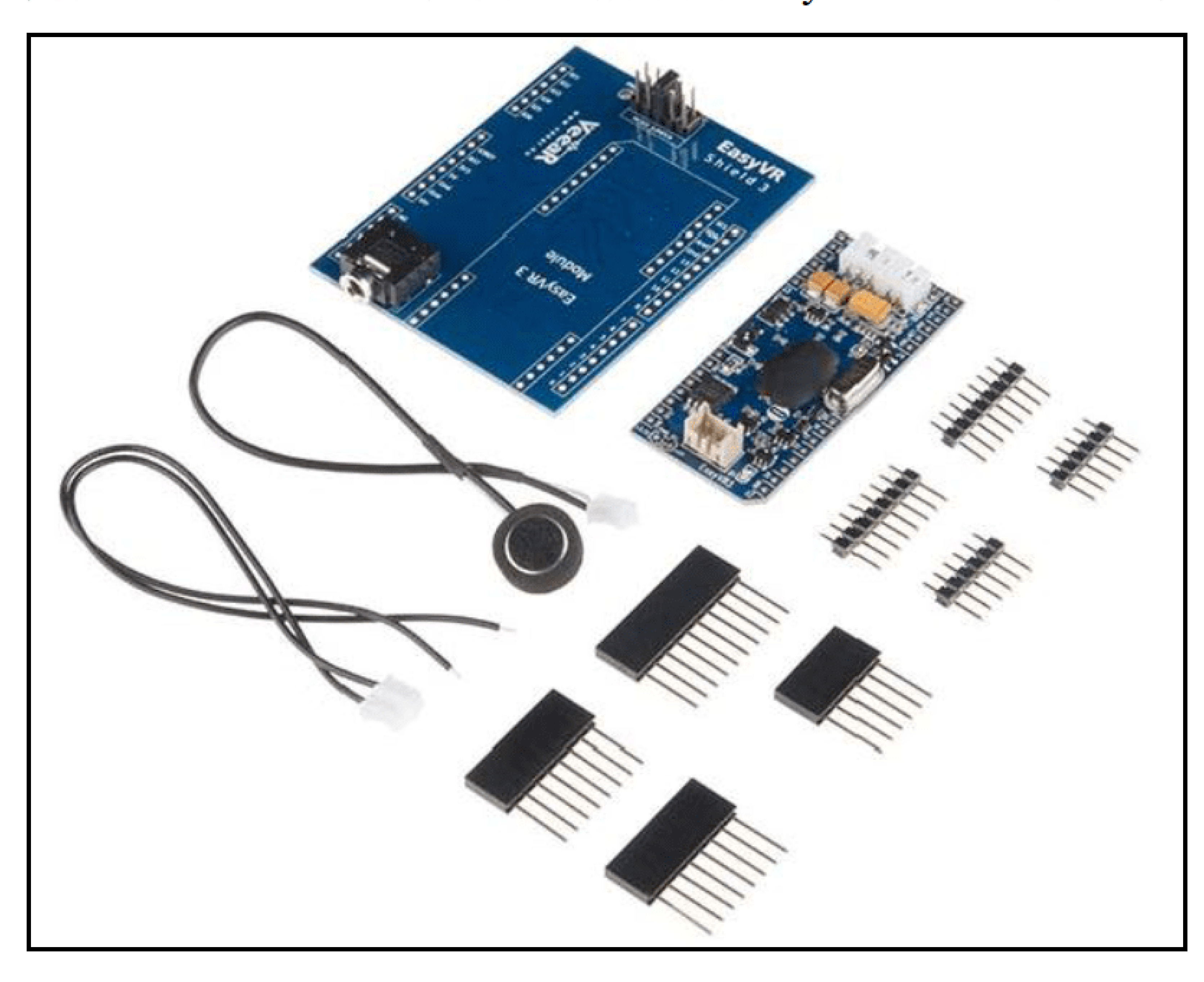


图 5-9

第二个模块是 Emic 2。它是 Parallax 设计的，用来协调 Grand Idea Studio（http://www.grandideastudio.com/），使得声音合成变得非常简单。可以通过串口协议发送文本给模块来生成人声。这个模块非常有用。可以访问 https://www.parallax.com/product/30016 获取更多信息或者购买。如图 5-10 所示是 Emic 2 模块的一个示例。

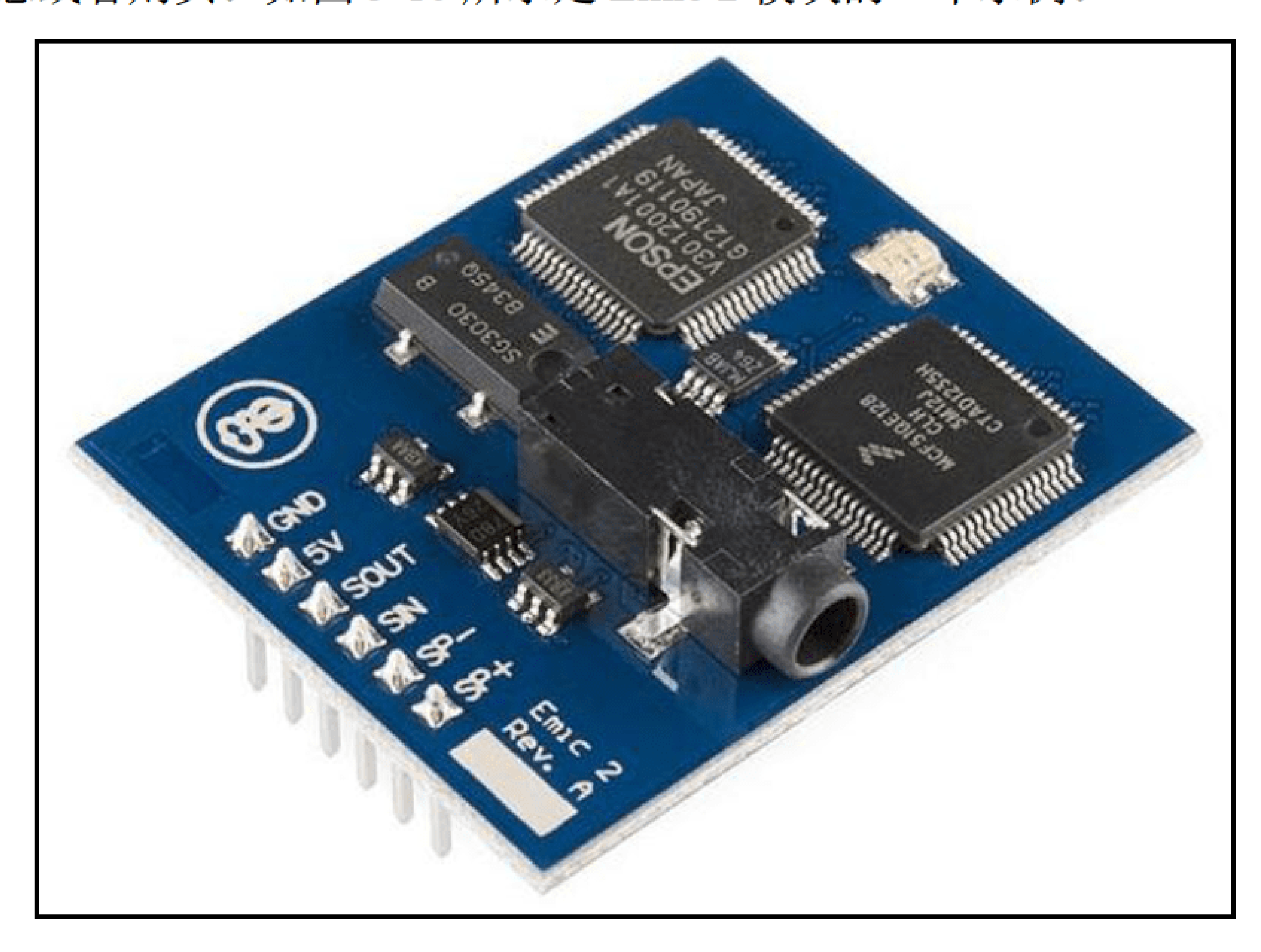


图 5-10

为物联网项目增加语音控制

设想如果可以通过语音命令开关灯，是不是非常酷？本节将使用 EasyVR 3 shield for Arduino 在 Arduino 上搭建一个非常简单的项目。

我们的应用场景就是利用语音命令控制开关 LED。或者也可以控制继电器或台灯。因为一些 EasyVR 工具只能在 Windows 平台工作，所以使用 Windows 操作系统作为整个例子的开发环境。这里用 Arduino Uno R3 板作为测试。

让我们开始吧！

设置 EasyVR shield 3

在项目中使用 EasyVR shield 3 之前，需要先安装一些软件和库。

首先下载 Arduino 的 EasyVR 库。可以从 https://github.com/RoboTech-srl/EasyVR-

Arduino/releases 下载。解压文件并放到 Arduino 库的文件夹下，或者选择菜单 Sketch | Include Library | Add .ZIP Library。选择 EasyVR 库的 ZIP 格式文件。

下一步是安装 EasyVR Commander 工具，网址为 http://www.veear.eu/downloads/，下载并按照要求安装。安装完毕会看到一些工具。后面的步骤中会用这些工具配置 EasyVR 模块，将在第 6 章进行介绍。

接着把 EasyVR shield 3 breakout 焊接到 Arduino 板上。在这个例子里使用的是 Arduino Uno R3。

如图 5-11 所示是 EasyVR shield 3 的设计图。

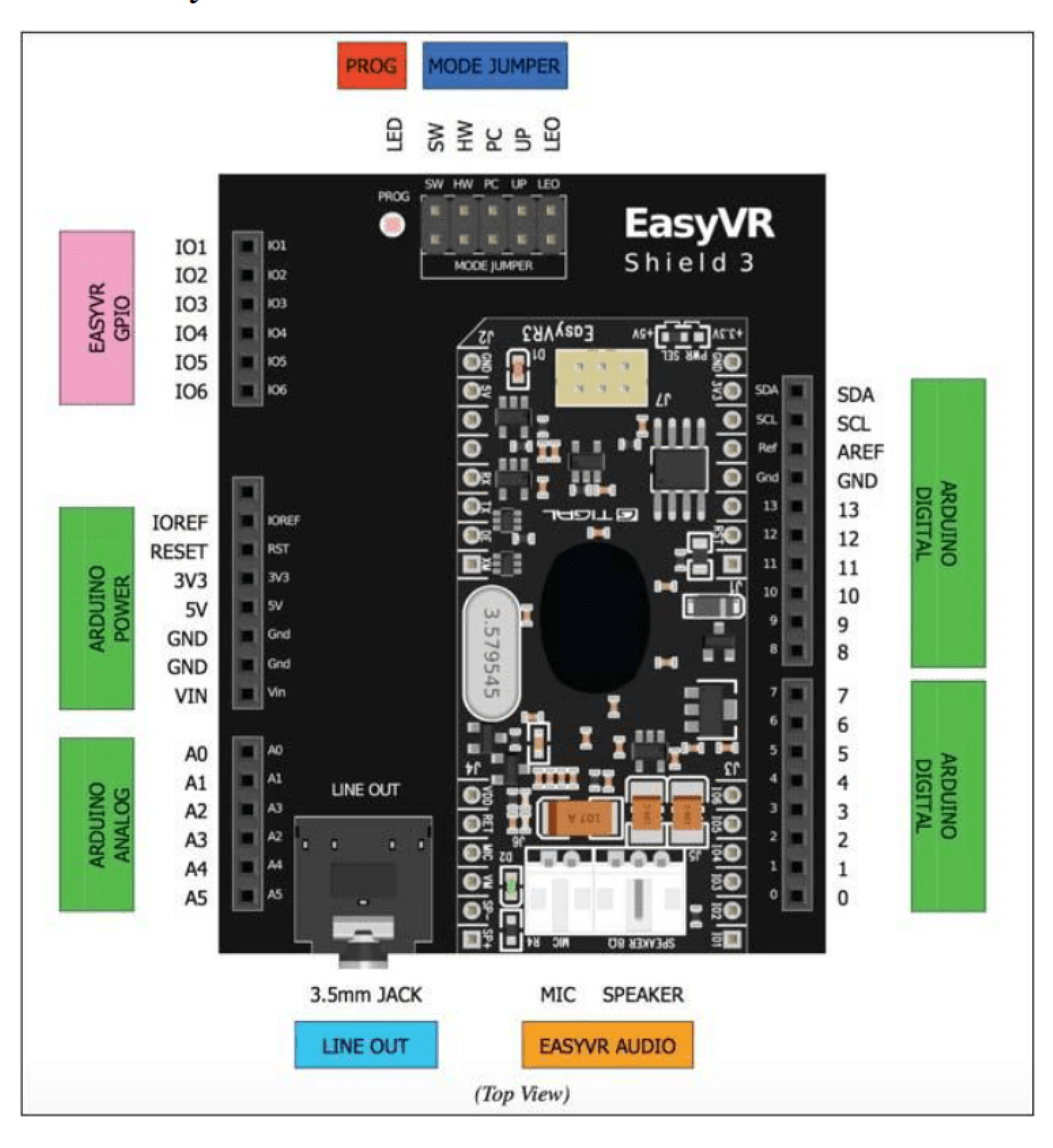


图 5-11

为了兼容 EasyVR 模块，把 MODE JUMPER 设置为 PC，然后通过 USB 把带着 EasyVR shield 的 Arduino 连接到计算机。

检查一下计算机是否检测到 Arduino 板。可以在 Windows 平台的设备管理器中的 Ports (COM & LPT)里看到，如图 5-12 所示。

现在打开 EasyVR Commander 工具。选择一个串口让带着 EasyVR 模块的 Arduino 连

接。如图 5-13 所示是一个已经连接上的例子。

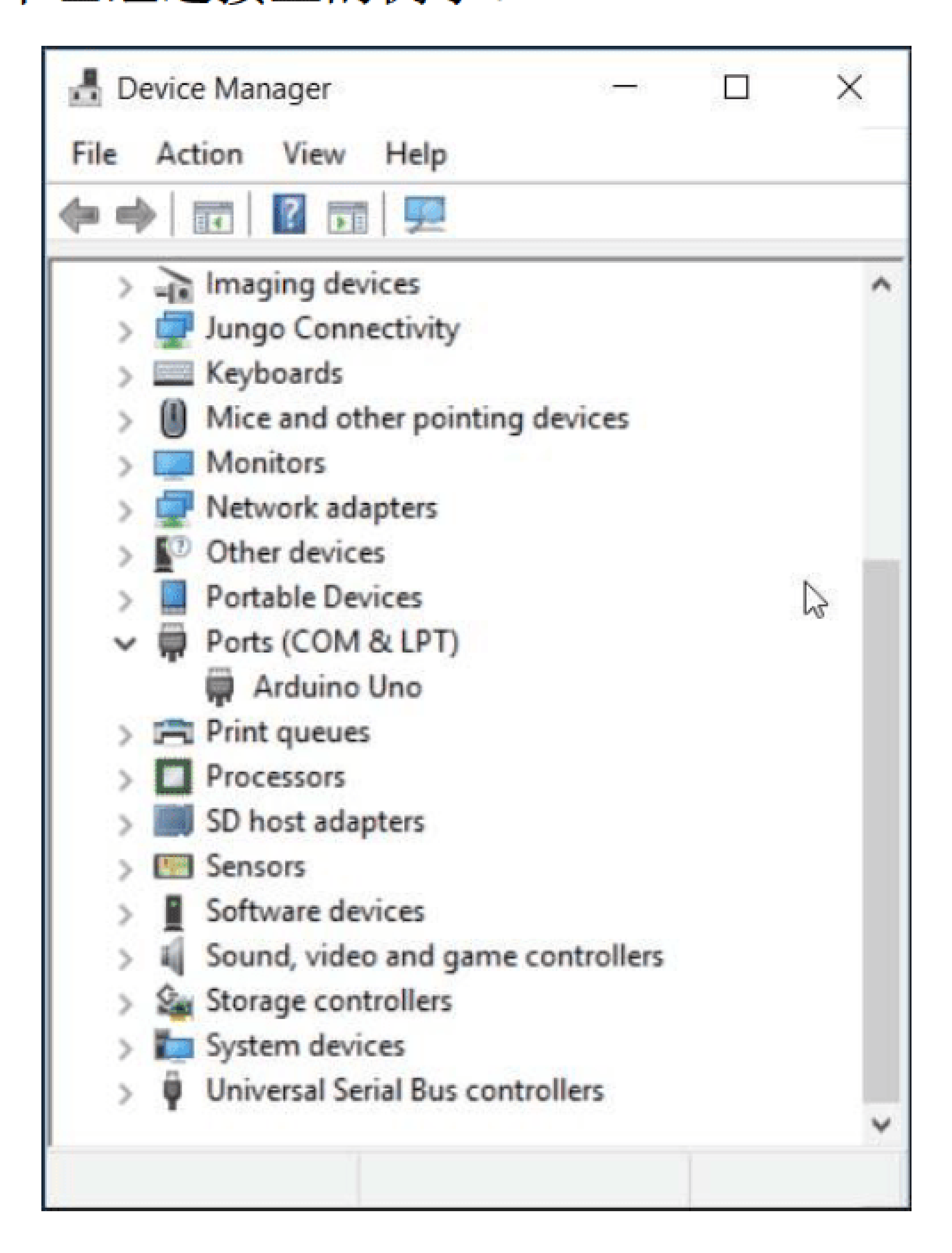


图 5-12

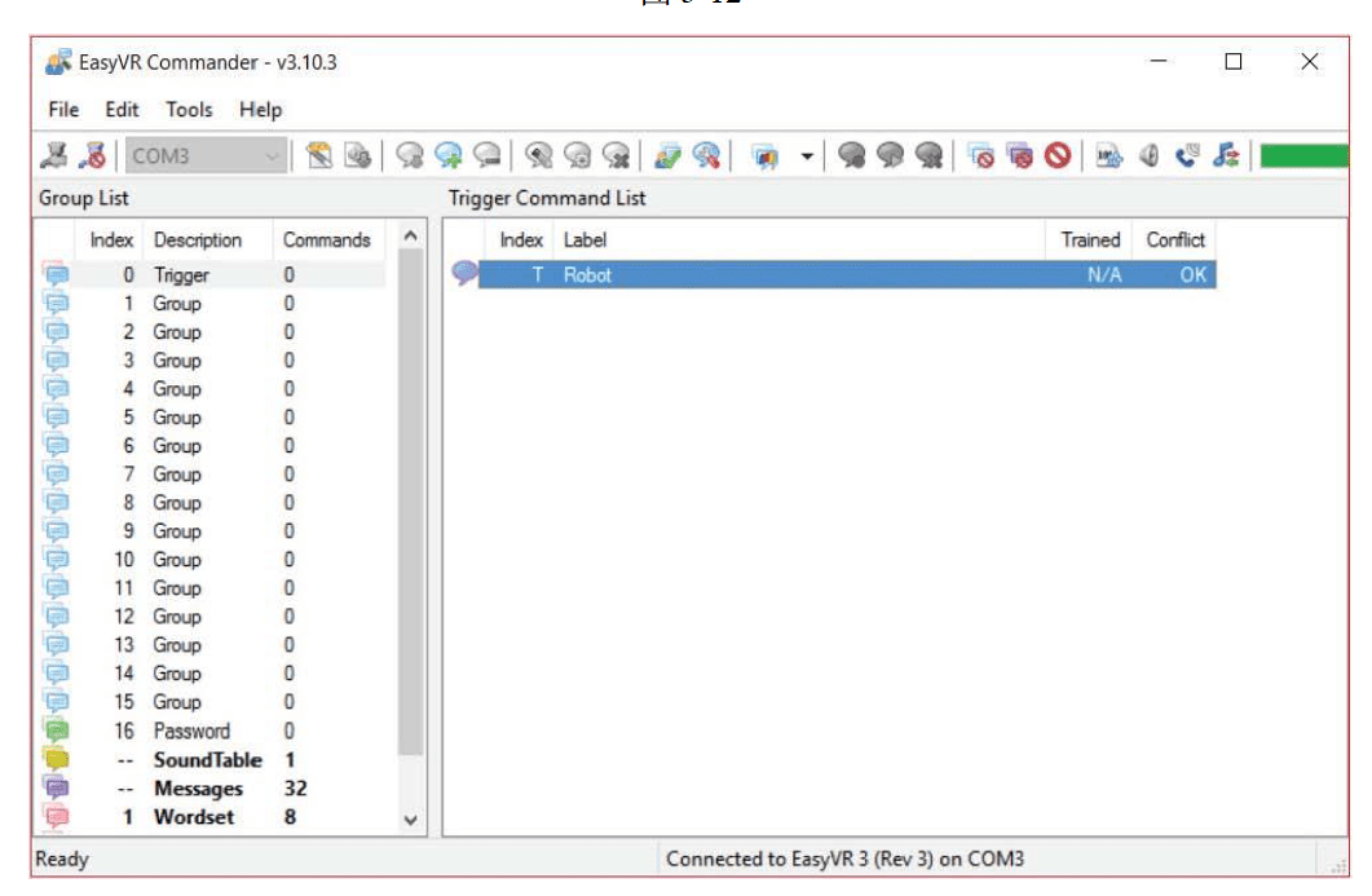


图 5-13

可以看到 index 0 到 16。Trigger 和 Password 是按照 group 的 index 保存。下面讲解如何使用 EasyVR Commander 工具。

创建语音命令

下面使用 EasyVR Commander 工具来创建如下语音命令模型：

- ARDUINO 代表启动程序。
- MYPASS 代表密码。
- LAMP_ON 代表打开 led。
- LAMP_OFF 代表关闭 led。
- LOGOUT 代表退出语音命令。

如图 5-14 所示是语音命令的状态图。

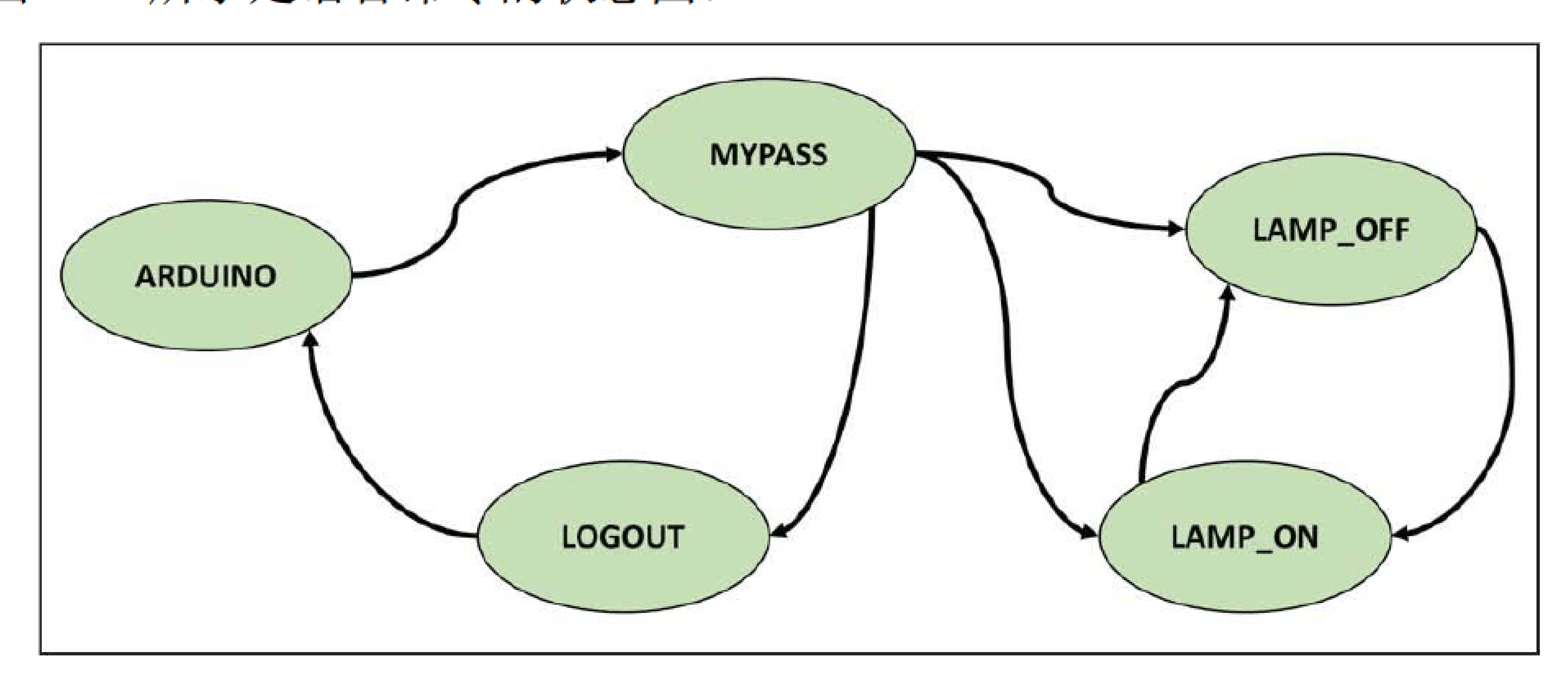


图 5-14

收到 ARDUINO 语音命令后 Arduino 开始接受命令，之后用户将被要求发送 MYPASS 命令。MYPASS 状态代表输入一个密码。如果 MYPASS 命令正确，用户可以通过 LAMP_ON 打开台灯或者通过 LAMP_OFF 关闭台灯。此外，用户可以使用 LOGOUT 退出。

如何运行？

连接 Arduino 到 EasyVR Commander 工具之后（确保 MODE JUMPER 在 PC 上已经设置），将看到图 5-15 的显示。

首先，创建 ARDUINO 语音命令。在 Group List 面板上选择 group 0（Trigger）。单击添加语音的标签图标。设置 ARDUINO 作为标签名字。

接着，训练和测试这个语音命令。选择 ARDUINO 标签并单击训练标签，你会看到如图 5-16 所示的对话框。

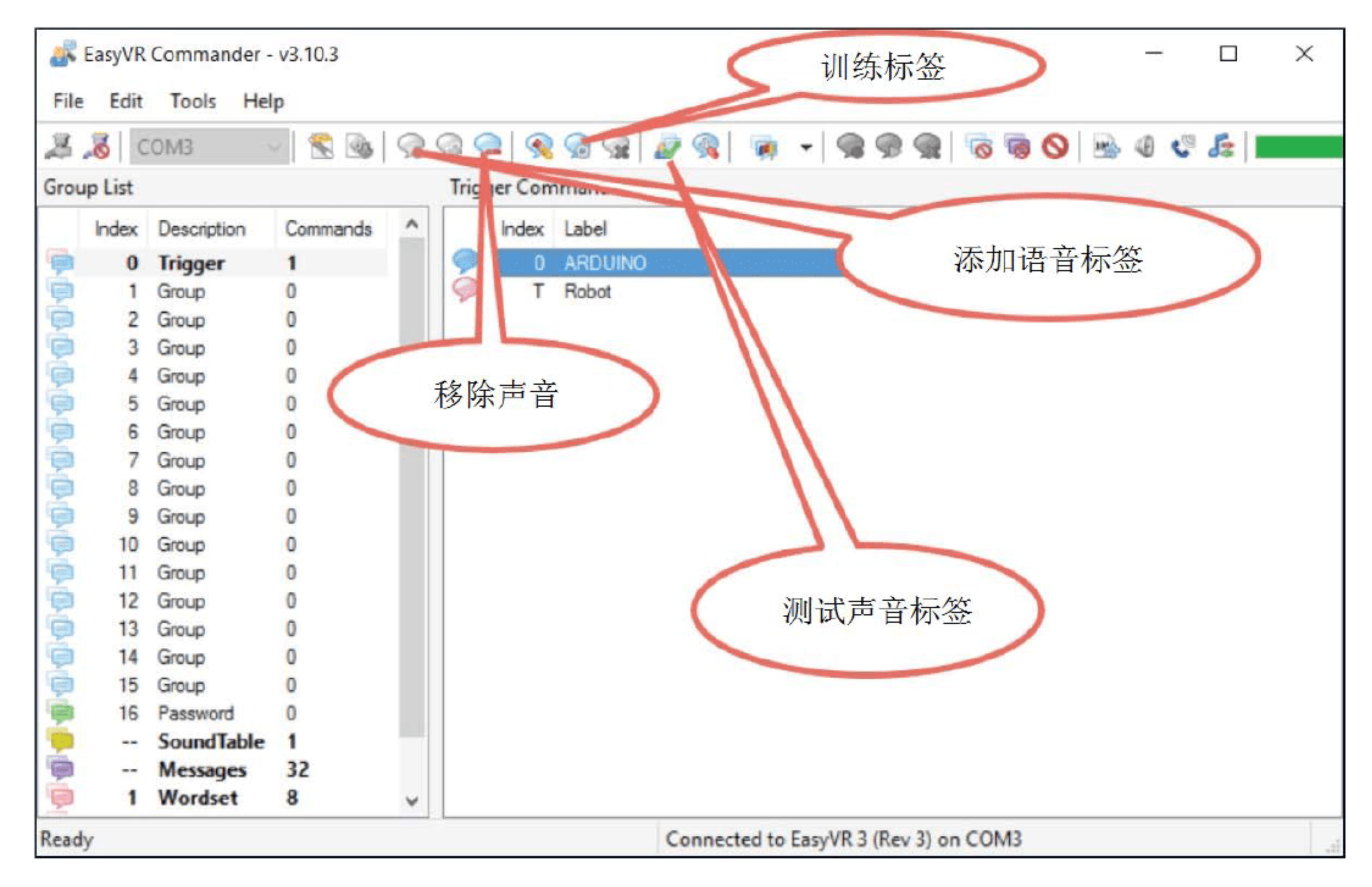

EasyVR Commander - v3.10.3
File Edit Tools Help
COM3
训练标签
Group List
添加语音标签
移除声音
测试声音标签
Index Description Commands
0 Trigger 1
1 Group 0
2 Group 0
3 Group 0
4 Group 0
5 Group 0
6 Group 0
7 Group 0
8 Group 0
9 Group 0
10 Group 0
11 Group 0
12 Group 0
13 Group 0
14 Group 0
15 Group 0
16 Password 0
-- SoundTable 1
-- Messages 32
1 Wordset 8
Index Label
0 ARDUINO
T Robot
Ready
Connected to EasyVR 3 (Rev 3) on COM3

图 5-15

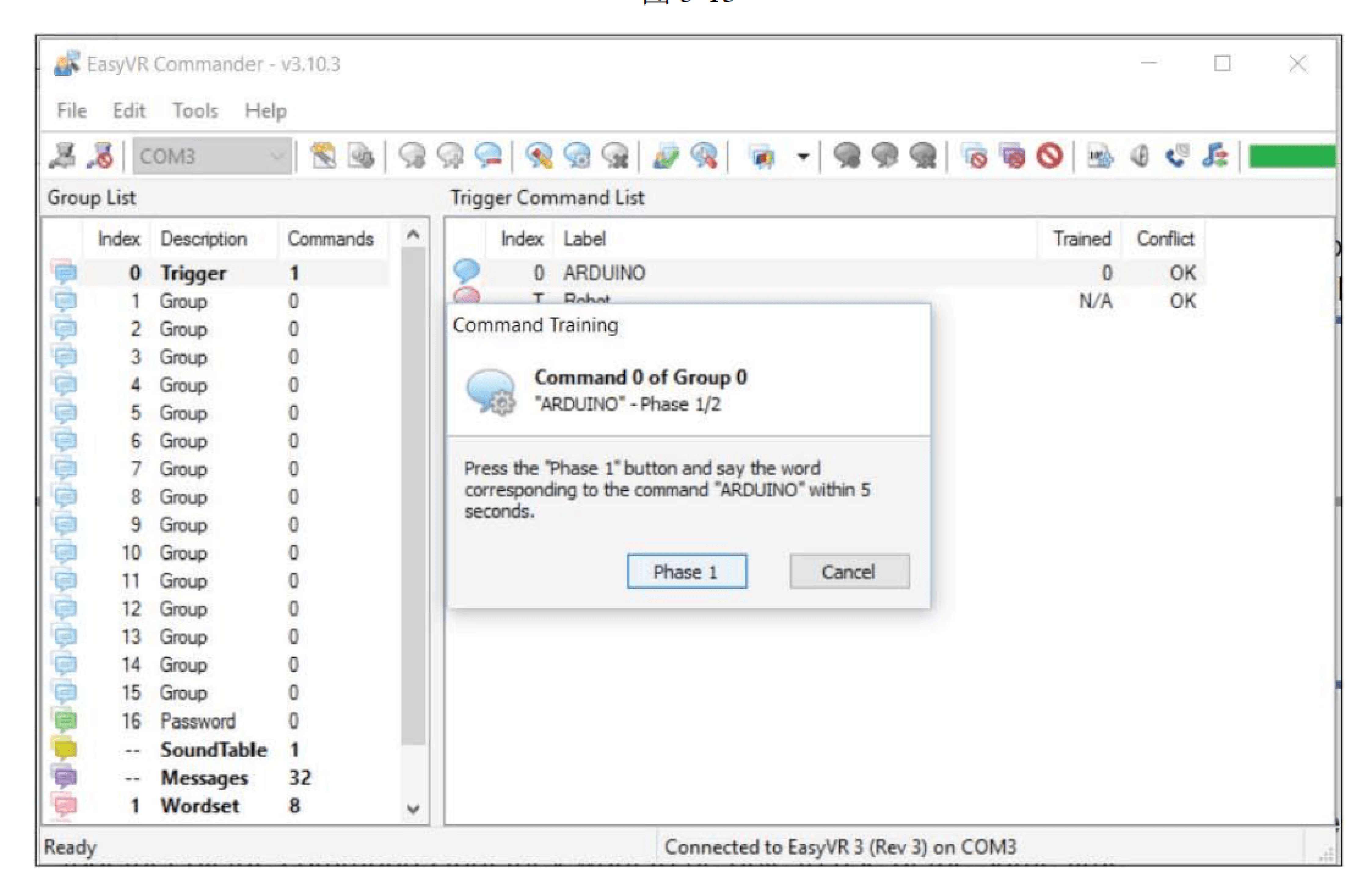

EasyVR Commander - v3.10.3
File Edit Tools Help
COM3
Group List
Trigger Command List
Index Description Commands
0 Trigger 1
1 Group 0
2 Group 0
3 Group 0
4 Group 0
5 Group 0
6 Group 0
7 Group 0
8 Group 0
9 Group 0
10 Group 0
11 Group 0
12 Group 0
13 Group 0
14 Group 0
15 Group 0
16 Password 0
-- SoundTable 1
-- Messages 32
1 Wordset 8
Index Label Trained Conflict
0 ARDUINO 0 OK
N/A OK
Command Training
Command 0 of Group 0
"ARDUINO" - Phase 1/2
Press the "Phase 1" button and say the word corresponding to the command "ARDUINO" within 5 seconds.
Phase 1
Cancel
Ready
Connected to EasyVR 3 (Rev 3) on COM3

图 5-16

程序提供两次训练。单击 Phase 1 并说点什么，如 ARDUINO，程序会记录声音，接着为了 ARDUINO 标签再训练一次。

现在可以测试语音命令。EasyVR 语音命令只能在单个 group 里检测语音命令。因此如果想要在 group 0 里测试，需要单击 Trigger（index 0 或 group 0）。

之后，单击测试语音的图标，会看到如图 5-17 所示的写着 Speak now...的对话框。

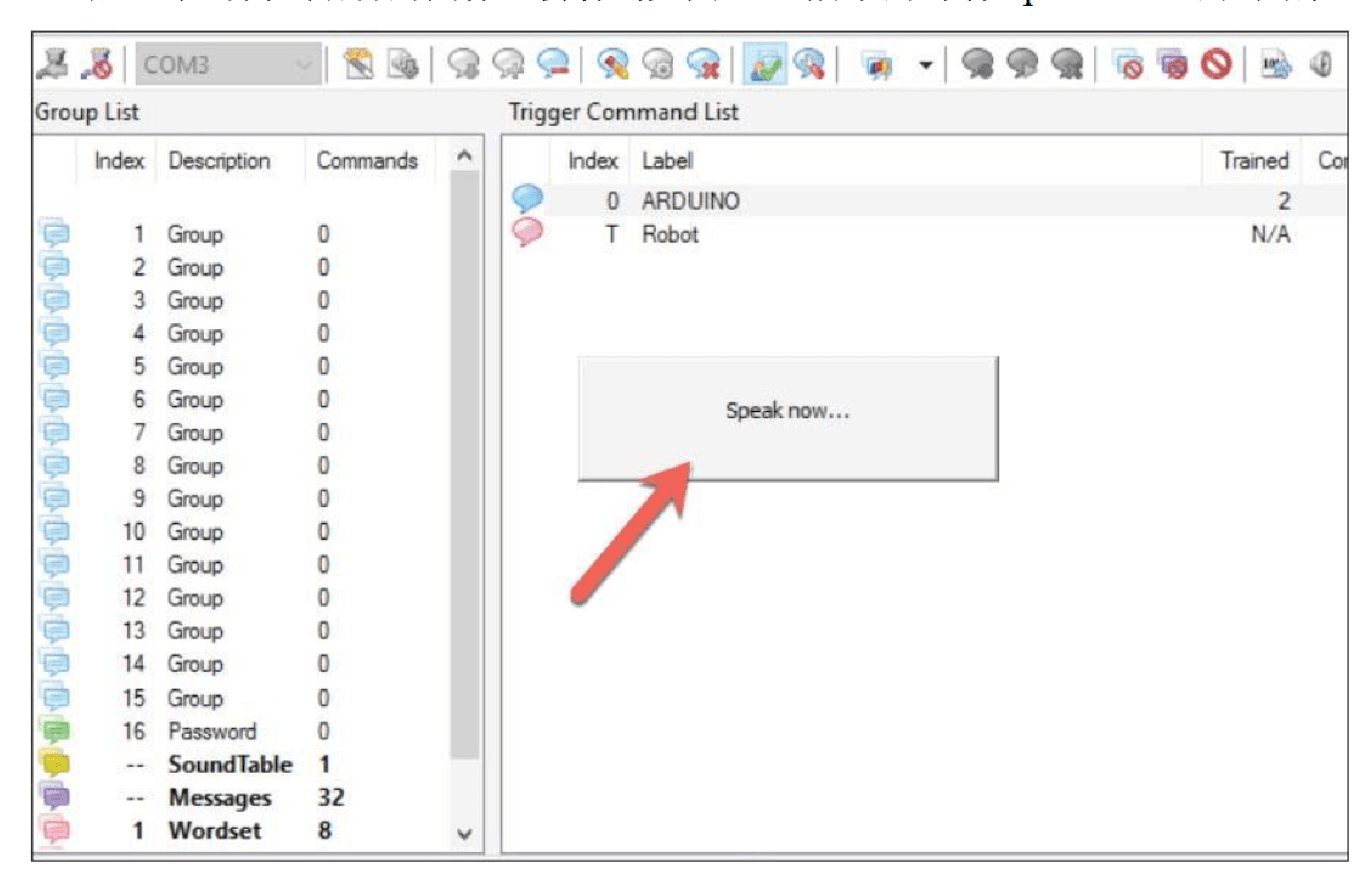


图 5-17

试着给 ARDUINO 标签输入一个声音。程序在检测到之后会高亮显示这个标签。

这是创建语音命令的完整流程。现在可以继续创建剩下的语音命令：

- MYPASS 标签在 Password group（index 16）。
- LAMP_ON 标签在 Group 1（index 1）。
- LAMP_OFF 标签在 Group 1（index 1）。
- LOGOUT 标签在 Group 1（index 1）。

可以试着创建一个新标签，训练并测试。

现在可以从定义好的标签上生成代码。在 EasyVR Commander 工具上选择菜单 File | Generate Code，会看到 sketch 代码文件（*.ino file）。在下一节将修改这个 Sketch 程序。

完成之后断开 Arduino 和 EasyVR commander 工具。继续编写 Sketch 程序。

给语音板布线

在这里使用一个 LED 灯。可以使用带着继电器的台灯。通过电阻器 330 Ohm 连接 LED 灯到数字 pin 8。如图 5-18 所示是焊线例子。

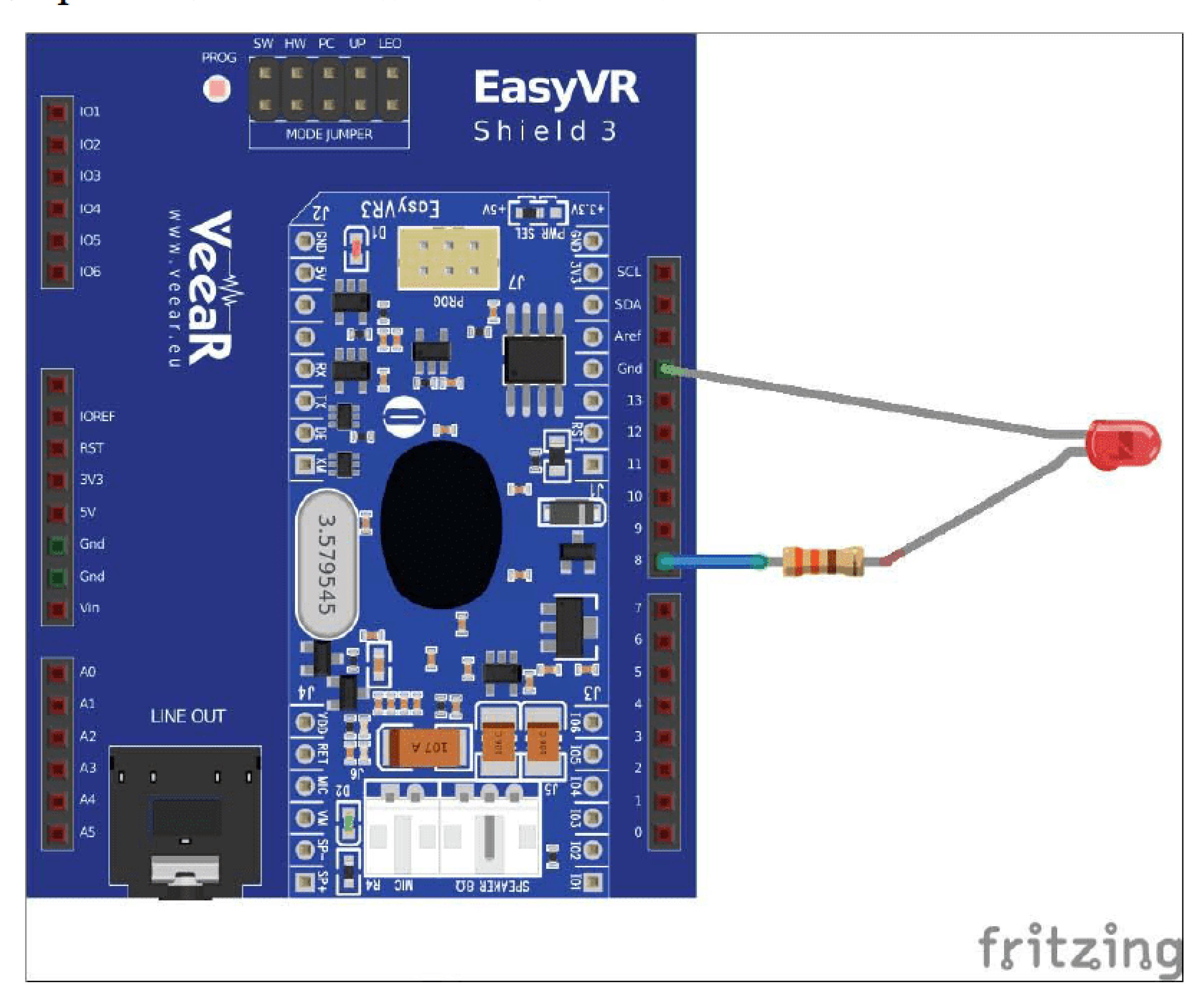


图 5-18

在 EasyVR shield 3 上更改 MODE JUMPER 为 SW 模式，这样就可以和 Arduino 及 EasyVR shield 3 通信。

编写 Sketch 程序

我们已经获得了 Sketch 程序的框架，现在只需修改程序。下面是修改的代码：

```
#include "Arduino.h"
#if !defined(SERIAL_PORT_MONITOR)
  #error "Arduino version not supported. Please update your IDE to  the
latest version."
#endif

#if defined(SERIAL_PORT_USBVIRTUAL)
```

```
  //Shield Jumper on HW (for Leonardo and Due)
  #define port SERIAL_PORT_HARDWARE
  #define pcSerial SERIAL_PORT_USBVIRTUAL
#else
  //Shield Jumper on SW (using pins 12/13 or 8/9 as RX/TX)
  #include "SoftwareSerial.h"
  SoftwareSerial port(12, 13);
  #define pcSerial SERIAL_PORT_MONITOR
#endif

#include "EasyVR.h"

EasyVR easyvr(port);

//Groups 和 Commands
enum Groups
{
  GROUP_0  = 0,
  GROUP_1  = 1,
  GROUP_16 = 16,
};

enum Group0
{
  G0_ARDUINO = 0,
};

enum Group1
{
  G1_LAMP_ON = 0,
  G1_LAMP_OFF = 1,
  G1_LOGOUT = 2,
};

enum Group16
{
  G16_MYPASS = 0,
};


int8_t group, idx;
```

```
int myled = 8;

void setup()
{
  //设置 PC 系列端口
  pcSerial.begin(9600);

  pinMode(myled, OUTPUT);

  //bridge mode?
  int mode = easyvr.bridgeRequested(pcSerial);
  switch (mode)
  {
  case EasyVR::BRIDGE NONE:
    //创建 EasyVR 串口
    port.begin(9600);
    //run normally
    pcSerial.println(F("---"));
    pcSerial.println(F("Bridge not started!"));
    break;

  case EasyVR::BRIDGE_NORMAL:
    //创建 EasyVR 串口（低速）
    port.begin(9600);
    //soft-connect the two serial ports (PC and EasyVR)
    easyvr.bridgeLoop(pcSerial);
    //resume normally if aborted
    pcSerial.println(F("---"));
    pcSerial.println(F("Bridge connection aborted!"));
    break;

  case EasyVR::BRIDGE BOOT:
    //创建 EasyVR 串口（高速）
    port.begin(115200);
    //软连接串口（PC 和 EasyVR）
    easyvr.bridgeLoop(pcSerial);
    //重新开始
    pcSerial.println(F("---"));
    pcSerial.println(F("Bridge connection aborted!"));
    break;
  }
```

```
  while (!easyvr.detect())
  {
    Serial.println("EasyVR not detected!");
    delay(1000);
  }

  easyvr.setPinOutput(EasyVR::IO1, LOW);
  Serial.println("EasyVR detected!");
  easyvr.setTimeout(5);
  easyvr.setLanguage(0);

  group = EasyVR::TRIGGER; //<-- start group (customize)
}

void action();

void loop()
{
  if (easyvr.getID() < EasyVR::EASYVR3)
    easyvr.setPinOutput(EasyVR::IO1, HIGH); //LED on (listening)

  Serial.print("Say username ");
  Serial.println(group);
  easyvr.recognizeCommand(group);

  do
  {
    //等待语音命令时可以做些其他处理
  }
  while (!easyvr.hasFinished());

  if (easyvr.getID() < EasyVR::EASYVR3)
    easyvr.setPinOutput(EasyVR::IO1, LOW); //LED关闭

  idx = easyvr.getWord();
  if (idx >= 0)
  {
    //built-in trigger (ROBOT)
    //group = GROUP_X; <-- 跳转到另一个 group X
    //group = GROUP_16;
```

```
    return;
  }
  idx = easyvr.getCommand();
  if (idx >= 0)
  {
    //打印 debug 信息
    uint8_t train = 0;
    char name[32];
    Serial.print("Command: ");
    Serial.print(idx);
    if (easyvr.dumpCommand(group, idx, name, train))
    {
      Serial.print(" = ");
      Serial.println(name);
    }
    else
      Serial.println();
    //发出声音
    easyvr.playSound(0, EasyVR::VOL_FULL);
    //执行一些动作
    action();
  }
  else //错误或者超时
  {
    if (easyvr.isTimeout())
      Serial.println("Timed out, try again...");
    int16_t err = easyvr.getError();
    if (err >= 0)
    {
      Serial.print("Error ");
      Serial.println(err, HEX);
    }
  }
}

void action()
{
    switch (group)
    {
    case GROUP_0:
      switch (idx)
```

```
      {
      case G0_ARDUINO:
        //编写动作代码
        Serial.println("Please say password");
        group = GROUP_16;
        break;
      }
      break;
    case GROUP_1:
      switch (idx)
      {
      case G1_LAMP_ON:
        digitalWrite(myled, HIGH);
        Serial.println("LAMP ON");
        break;
      case G1_LAMP_OFF:
        digitalWrite(myled, LOW);
        Serial.println("LAMP OFF");
        break;
      case G1_LOGOUT:
        group = EasyVR::TRIGGER;
        Serial.println("Logout");
        break;
      }
      break;
    case GROUP_16:
      switch (idx)
      {
      case G16_MYPASS:
        //编写动作代码
        Serial.println("OK.Now I'm waiting your command");
        group = GROUP_1;
        break;
      }
      break;
    }
}
```

程序基于状态图。

注意：记住 EasyVR 只能在激活的 group 上识别声音。

保存程序并上传到 Arduino 板。

测试

现在可以测试程序。试着发出为 ARDUINO 标签设置的声音，然后发出为 MYPASS 标签设置的声音。

之后，测试打开和关闭 LED 灯的语音命令。

最后测试 LOGOUT 标签的语音，之后 Arduino 只会等待 ARDUINO 标签的语音。

让 IoT 板说话

这非常有趣，想象一下 IoT 板发出一个语音的通知，告诉周围的环境的变化，如温度湿度等。只要实现了文本到语音的功能即可。

在这个例子里，使用 Emic 2 for Text-To-Speech 模块和 Arduino Uno R3 板。可以从 Sparkfun 商店购买到，网址为 https://www.sparkfun.com/products/11711。

让我们开始吧！

设置

这里使用 EMIC2 库用在 Arduino Sketch 和 EMIC 2 模块之间通信。可以在 https://github.com/pAIgn10/EMIC2 中找到。下载并将其放到 Arduino 库文件夹下。

现在可以使用 Arduino IDE 来编写 Sketch 程序。

布线

连接 EMIC 2 模块到 Arduino 板非常容易。连线如下：

- 连接 EMIC2 5V 到 Arduino 5V。
- 连接 EMIC2 GND 到 Arduino GND。
- 连接 EMIC2 SOUT 到 Arduino digital pin 9。
- 连接 EMIC2 SIN 到 Arduino digital pin 8。
- 连接 EMIC2 SP-到 speaker -。
- 连接 EMIC2 SP+到 speaker +。

连线的例子如图 5-19 所示。

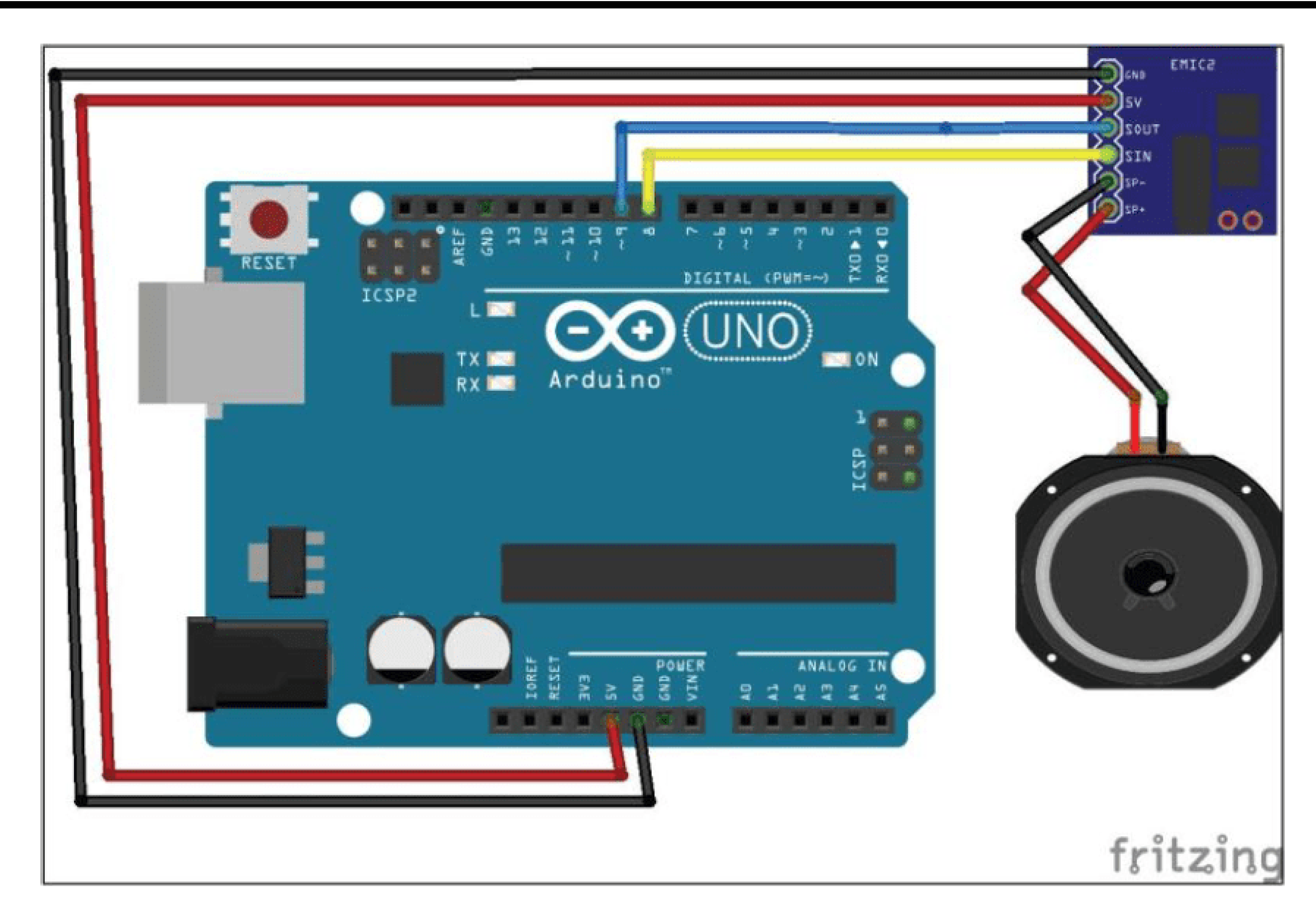


图 5-19

编写 Sketch 程序

可以使用 EMIC2 库的 emic.speak()来实现文本到语音的程序。为了方便测试，此处写了一个简单的程序来说出“Hello Arduino”和“I am waiting your command”这两句话。

编写如下的 Sketch 程序：

```
#include <SoftwareSerial.h>
#include "EMIC2.h"

//参见 http://arduino.cc/en/Reference/SoftwareSerial
#define RX_PIN 9  //连接 SOUT Emic 2 模块到 RX pin
#define TX_PIN 8  //连接 SIN Emic 2 模块到 TX pin

EMIC2 emic;

void setup() {
  emic.begin(RX_PIN, TX_PIN);
  emic.setVoice(8);  //设置音量（9 个选择：0～8）

}

void loop() {
  //把主要的代码放这里并重复运行：
```

```
  emic.setVolume(10);
  emic.speak("Hello Arduino");
  emic.resetVolume();
  delay(2000);

  emic.speak("I am waiting your command");
  delay(3000);
}
```

保存程序为 ArTTSDemo。

测试

编译并上传 Sketch 程序到 Arduino 板。可以听到刚才所设置的两句话。

这个例子很简单。可以在此基础上扩展程序，如从网上读取数据，然后让机器读出这些词，或者让 Arduino 报出当前的温度和湿度。

让 Raspberry Pi 说话

在之前的章节，已经学会如何在 Arduino 板上集成语音模块，本节将学习如何在软件中处理语音。

下面将使用 Raspberry Pi 来作为例子学习如何处理歌曲。这里使用的是 Raspberry Pi 3 和 Raspbian Jessie 系统，也可以使用最新的 Raspbian 系统。

设置

一些 Raspberry Pi 模块，如 Raspberry Pi 3，通过音频插座提供音频输出，如图 5-20 所示。

可以连接头戴式耳机、耳塞式耳机或者外部扬声器到 Raspberry Pi 板上。作为测试，此处使用外部扬声器——JBL 扬声器。

为了兼容外部扬声器，需要设置一下 Raspberry Pi。可以在终端里使用 rasp-config 命令。命令如下：

```
$ sudo raspi-config
```

然后就可以看到 rasp-config 的窗口。选择第 9 行——Advanced Options，如图 5-21 所示。

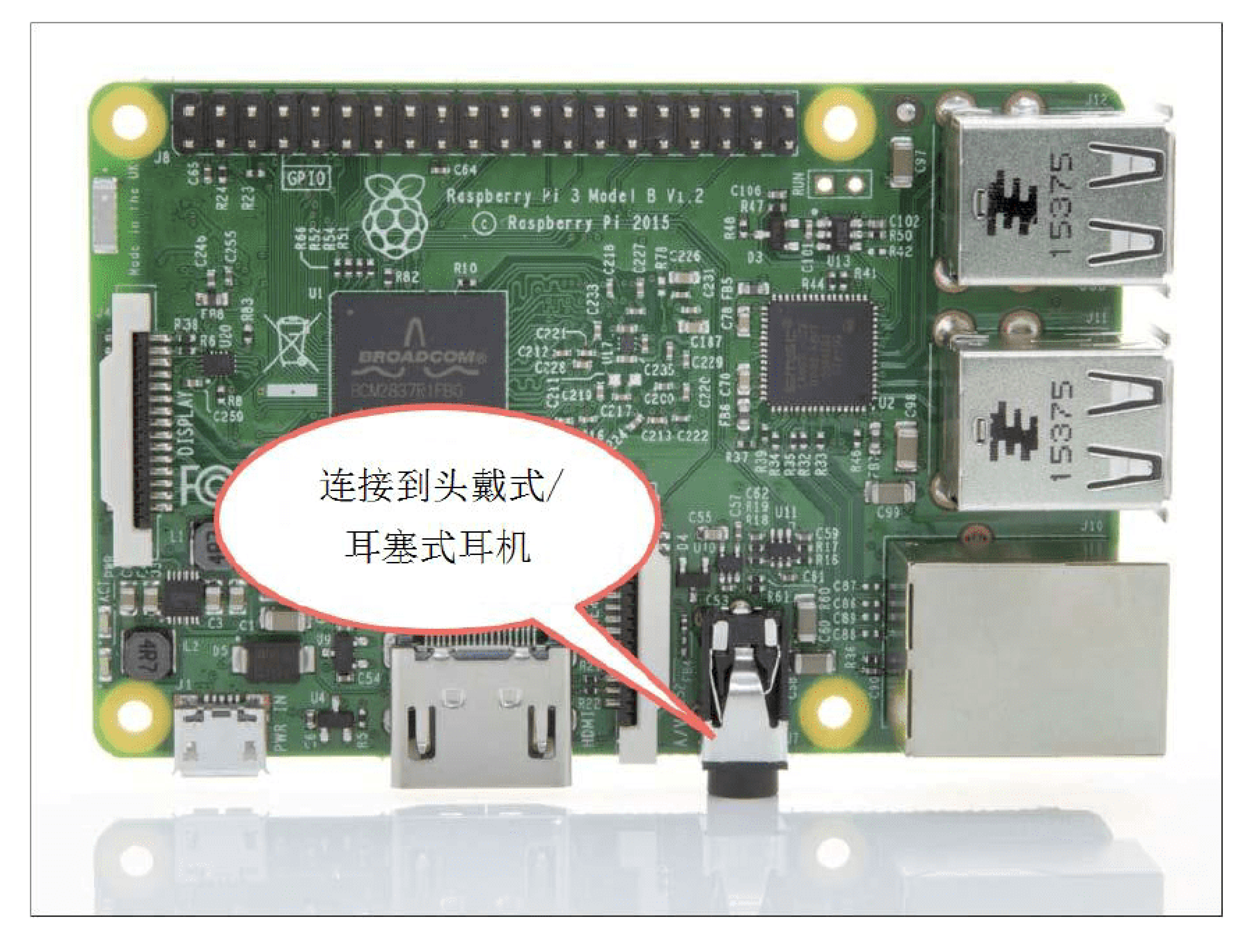


图 5-20

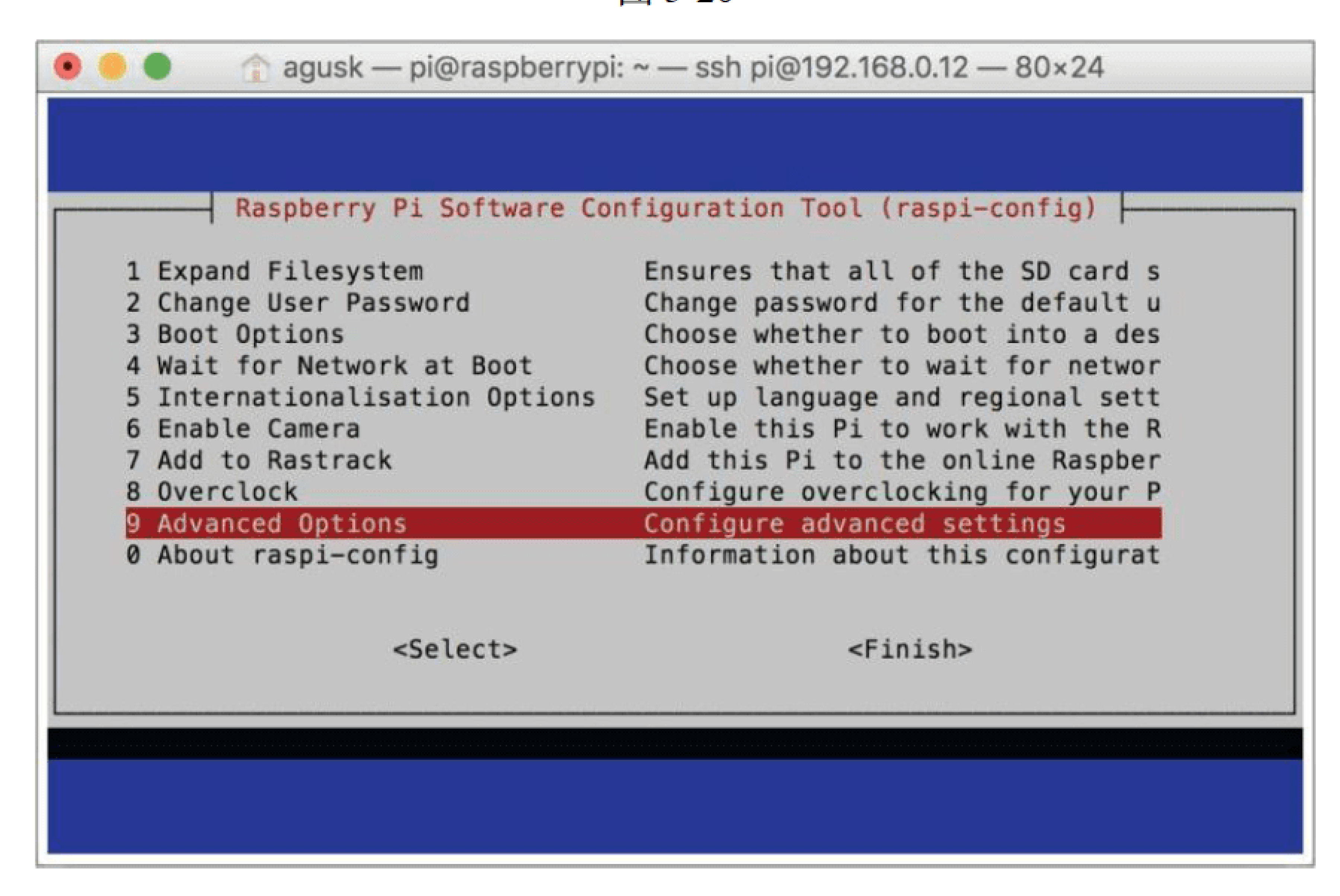


图 5-21

接着选择第 9 行的 A9 Audio 来配置 Raspberry Pi 的音频，如图 5-22 所示。

接着可以看到音频输出选项的列表。选择第二行的 1 Force 3.5mm ('headphone') jack，如图 5-23 所示。

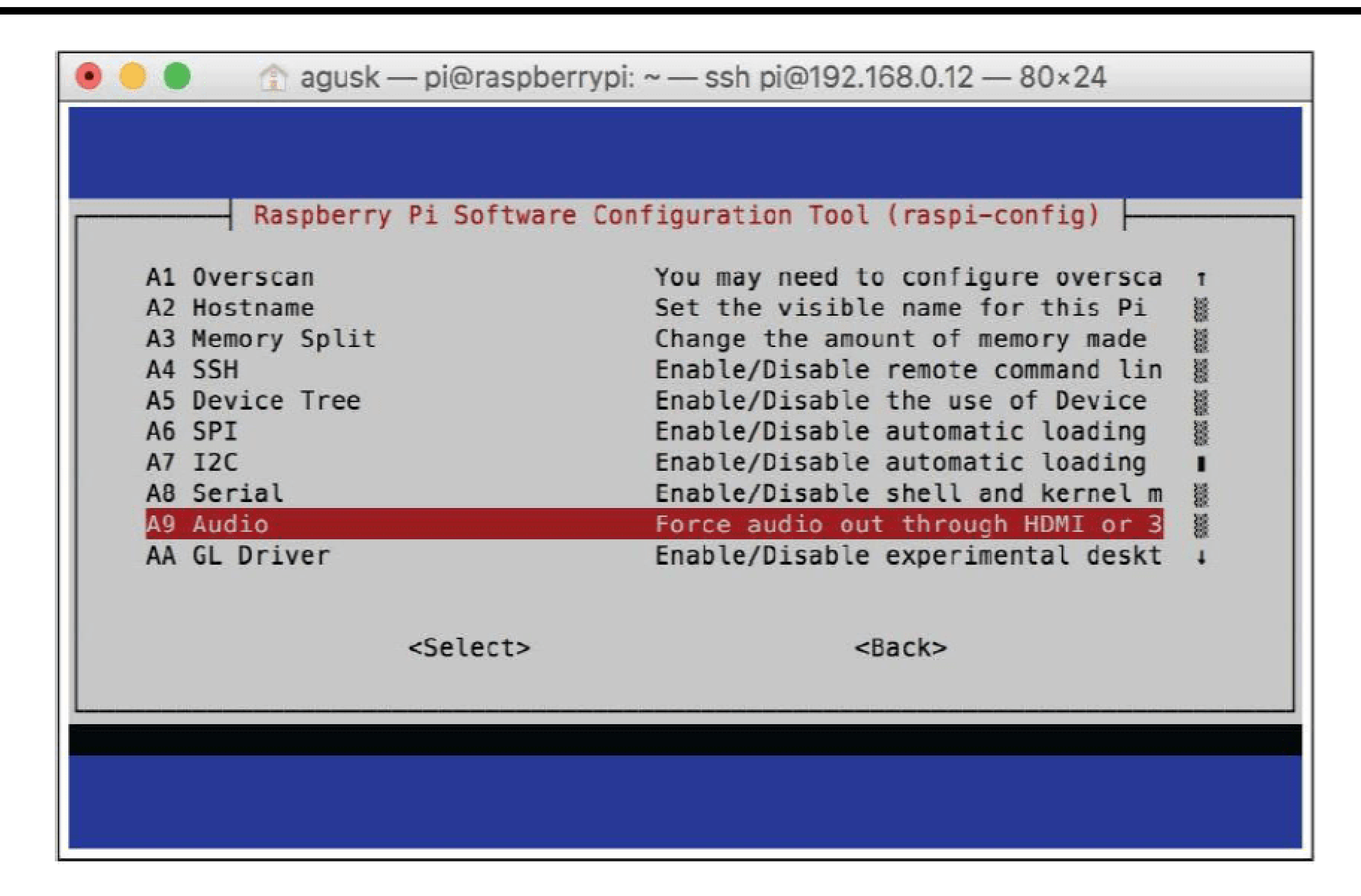


图 5-22

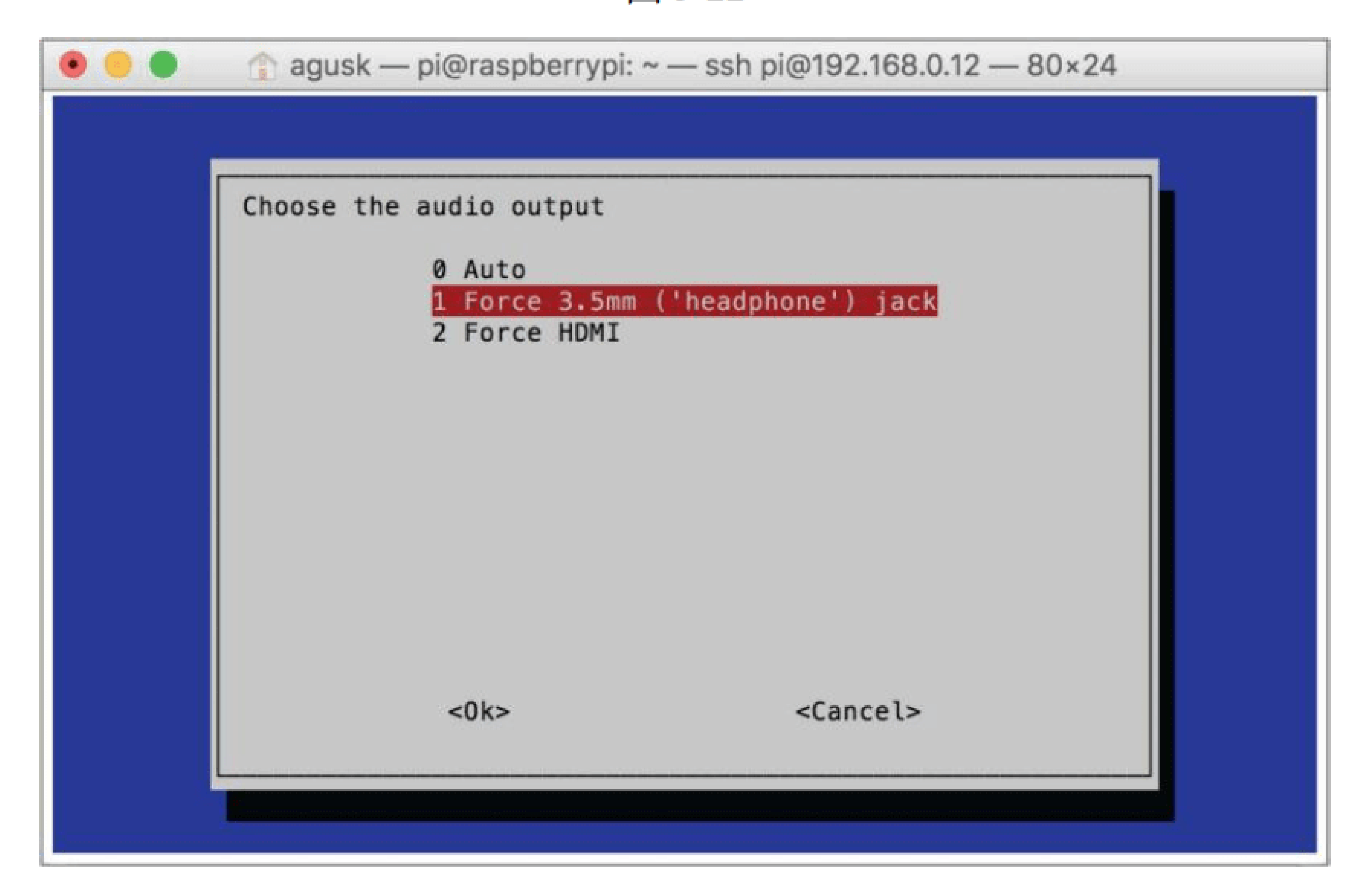


图 5-23

现在耳机音频输出已经配置好了。下一步是安装 audio 库。这里使用 festival 库 http://www.cstr.ed.ac.uk/projects/festival/。

在 Raspberry Pi 终端上输入如下两个命令来安装 festival 库：

```
$ sudo apt-get update
$ sudo apt-get install festival
```

请确保 Raspberry Pi 已经联网。

festival 库也提供一些语音模型，可以在 https://packages.debian.org/jessie/festival-voice 网站上查看并安装一些额外的声音模型。

完成后，来测试生成语音“Good morning!”。可以在终端上输入如下命令：

```
echo "Good morning!" | festival --tts
```

可以听到报出 Good morning 的声音！

编写 Python 程序

接着使用 festival 库。在本节中，编写 Python 程序访问 festival 库，这样可以定制程序并且结合 GPIO 编程。幸运的是，可以使用 festival 库的 Python 版 pyfestival 库。详细内容见 https://github.com/techiaith/pyfestival。

使用如下命令在 Raspberry Pi 上安装 pyfestival 库：

```
$ sudo apt-get install python python-dev festival festival-dev
$ sudo pip install pyfestival
```

在 pyfestival 库中调用 festival.sayText()函数生成人声。调用 festival.sayFile()来从文件中生成人声。

编写如下脚本：

```
import festival

festival.sayText("I am Raspberry Pi")
festival.sayFile("ch05_test_tts.txt")
```

保存文件为 ch05_tts.py。

可以在 ch05_test_tts.txt 文件里写任何内容。如下是一个例子：

```
The Raspberry Pi is a series of credit card-sized single-board computers
developed in the United Kingdom by the Raspberry Pi Foundation to promote
the teaching of basic computer science in schools and developing
countries.
```

保存文本。

运行如下命令测试 ch05_tts.py 文件：

```
$ python ch05_tts.py
```

可以从耳机或者扬声器里听到声音。

下一步是什么？

这是一个非常简单的程序，可以结合传感器、驱动器或者其他电子部件来扩展它。例如，可以实现一个按下 Raspberry Pi 上的按钮程序就说话的功能。

还可以结合之前学习的语音命令，这样 Raspberry Pi 就可以和你用语音交流了。

总　结

在本章中，学习了一些基本的声音和语音的处理，也探索了如何集成一些声音和语音的模块到物联网项目中。首先编写了读取声音强度的程序，然后利用 EasyVR 和 EMIC 2 模块实现语音命令和文本到语音的程序。最后，还让 Raspberry Pi 能够说话。

在第 6 章，将学习如何在 IoT 项目里集成云技术作为后端计算中心。

引　用

下面是一些推荐的论文、数据和网站，可以了解更多关于本章的内容。

1. Juang, B. H.; Rabiner, Lawrence R. *Automatic speech recognition–a brief history of the technology development*. http://www.ece.ucsb.edu/faculty/Rabiner/ece259/Reprints/354_LALI-ASRHistory-final-10-8.pdf.

2. Benesty, Jacob; Sondhi, M. M.; Huang, Yiteng. Springer Handbook of Speech Processing. Springer Science & Business Media.2008.

3. Wu Chou, Biing-Hwang Juang. Pattern Recognition in Speech and Language Processing. CRC Press, 2003.

第 6 章　为物联网项目搭建数据云

在本章中，将探索如何在 IoT 项目里应用云平台。云平台所代表的后端基础建设是非常重要的。把 IoT 板分发到世界不同的地方需要对传感器的数据更加注意，因此拥有一个具备数据科学服务的云平台对于扩展项目非常关键。

本章分为如下几个主题：

- 介绍云平台技术
- 介绍基于云的数据科学
- 连接 IoT 板到云服务器
- 搭建科学型数据云
- 构建基于科学型数据云的 IoT 应用

对云技术的介绍

云技术是指将本地的计算和数据移到网络中其他服务器上。投资大量的运算基础设施是非常昂贵的开销，除非你非常清楚这个风险。

通常来说，云技术提供了一个动态的基础设施，这样在需要升级设施时就不用停止系统。云技术有 3 个关键词：SaaS、PaaS 和 IaaS。

SaaS（软件即服务）是云平台消费者最熟悉的形式。SaaS 把管理软件和部署软件的任务交给了第三方的服务。SaaS 的例子包括 DropBox、Google Apps 和一些其他存储解决方案。

PaaS（平台即服务）提供了软件开发和部署的平台。消费者只需要关注他们的业务部分。可以选择最适合业务的平台而不用考虑网络和服务器这些基础设施。PaaS 的例子包括 Heroku、Google App Engine 和 Red Hat's OpenShift。

IaaS（基础设施即服务）通过面板或 API 提供云服务器和相关的资源。如果不想管理这些硬件资产，或者不想投资在一个数据中心上，这会很有帮助。IaaS 通常提供一个可扩展的云计算模型，同时允许自动部署服务器、处理电源、存储和网络。

一些云平台供应商，如 AWS IoT 和 Azure IoT，提供 SDK/API 以帮助 IoT 板子连接到他们的服务。可以在云服务器上管理和分析传感数据，如图 6-1 所示。

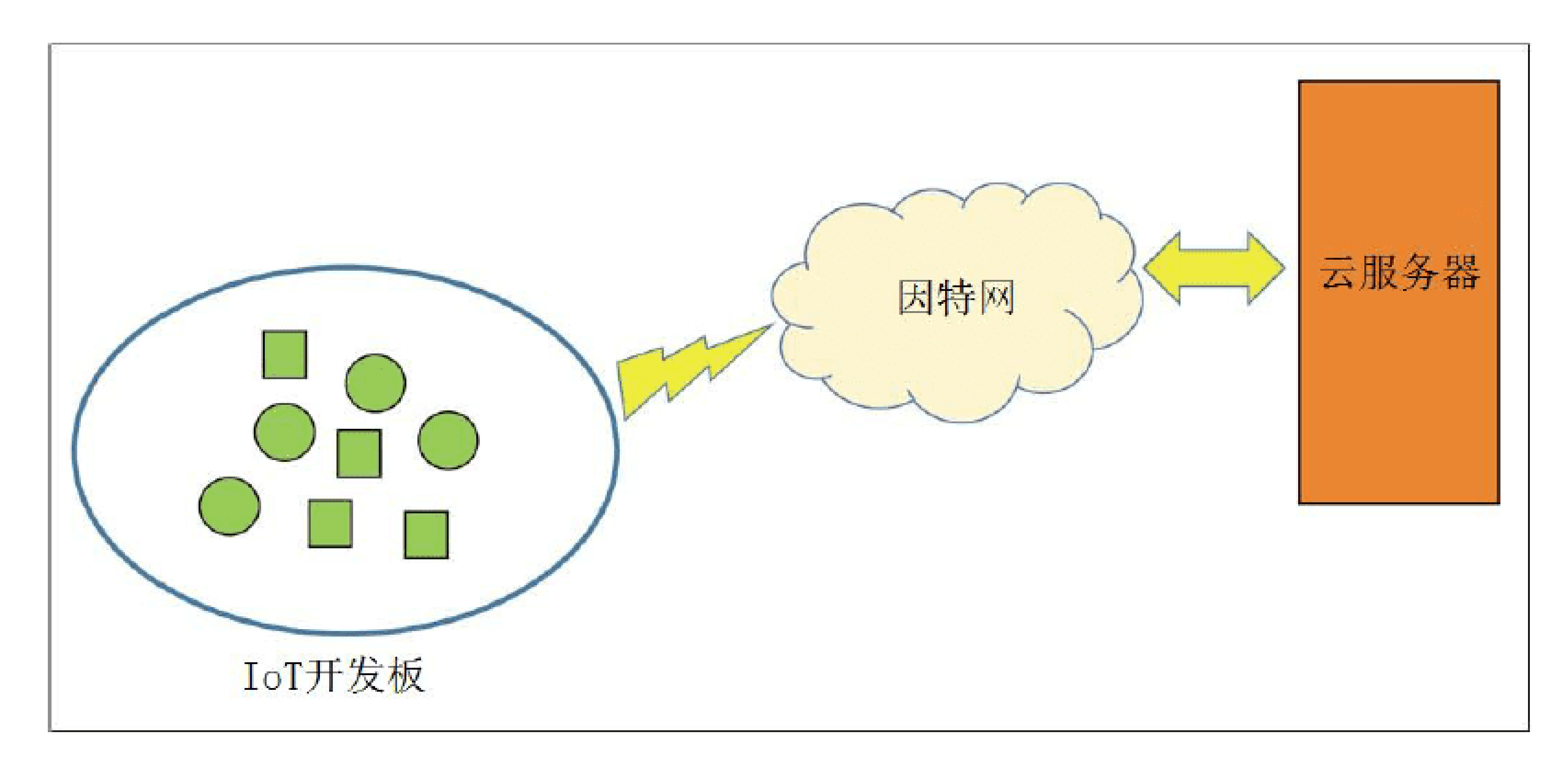


图 6-1

介绍基于云的数据科学

在数据科学领域，讨论回归、分类和预测问题。它们需要大量的计算资源来执行对应的数据科学任务。

基于云的数据科学是一个解决资源问题的方案。可以充分优化资源来执行分类预测任务。一个高可用的服务通常是被云平台供应商提供的。

一些大公司，如微软、亚马逊和谷歌已经实现了自己的数据科学服务器。可以使用 R 或者 Python 为数据科学编写代码。它们负责在预处理和后处理阶段处理数据。

图 6-2 所示是一个基于云的机器学习面板，即微软的 Azure 机器学习面板。

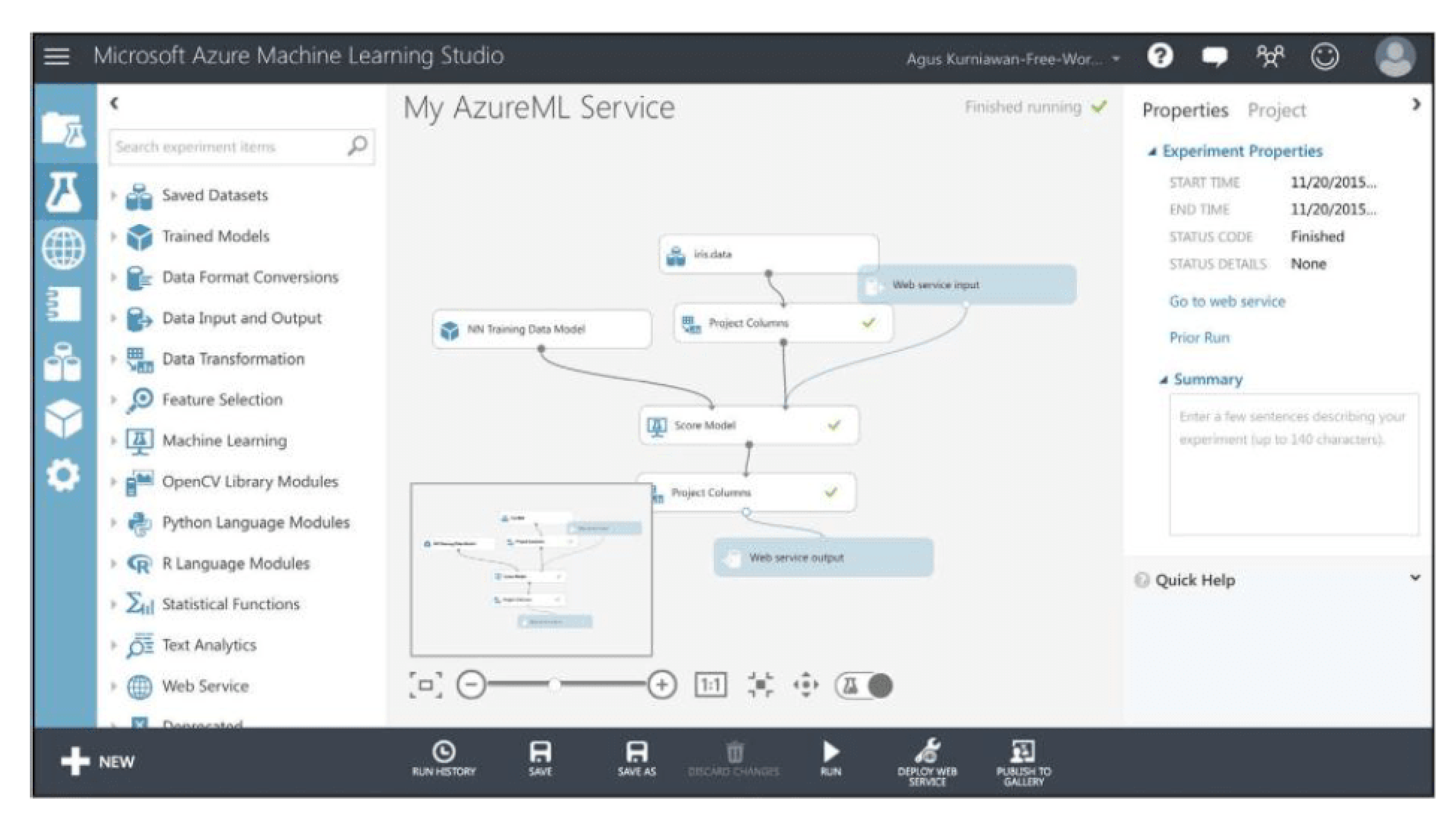


图 6-2

连接 IoT 板到云服务器

连接到云服务器意味着 IoT 板需要有网络功能。可以在 IoT 板上使用 Ethernet 模块或者无线模块。

有许多可以在 IoT 板上集成的云平台。本节将一起探索一些云平台的使用。

让我们开始吧！

微软 Azure IoT

微软从 Azure 开始它的云服务，为解决 IT 问题提供了很多选项。微软提供 Azure 入口来管理云服务，Azure 入口如图 6-3 所示。

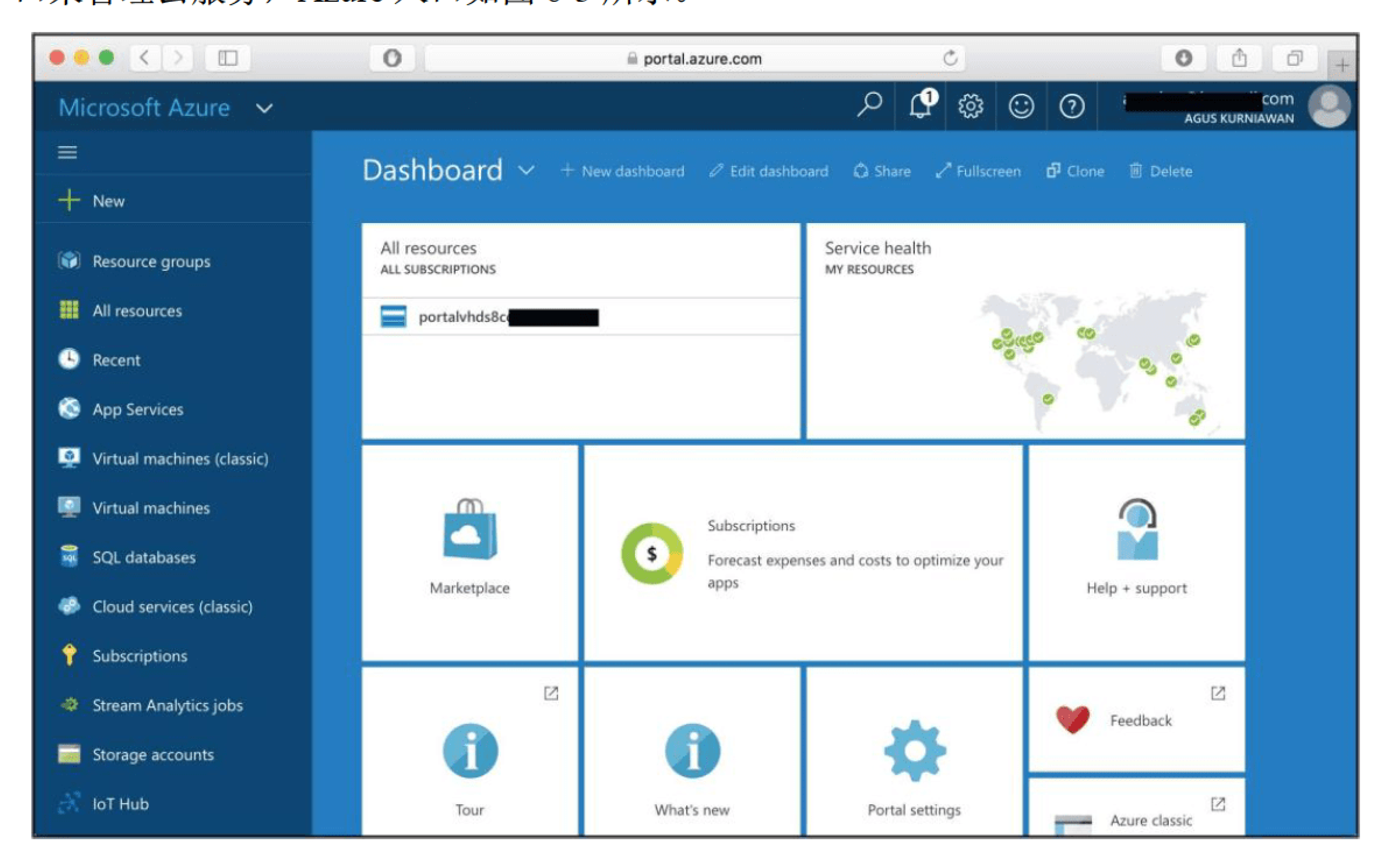


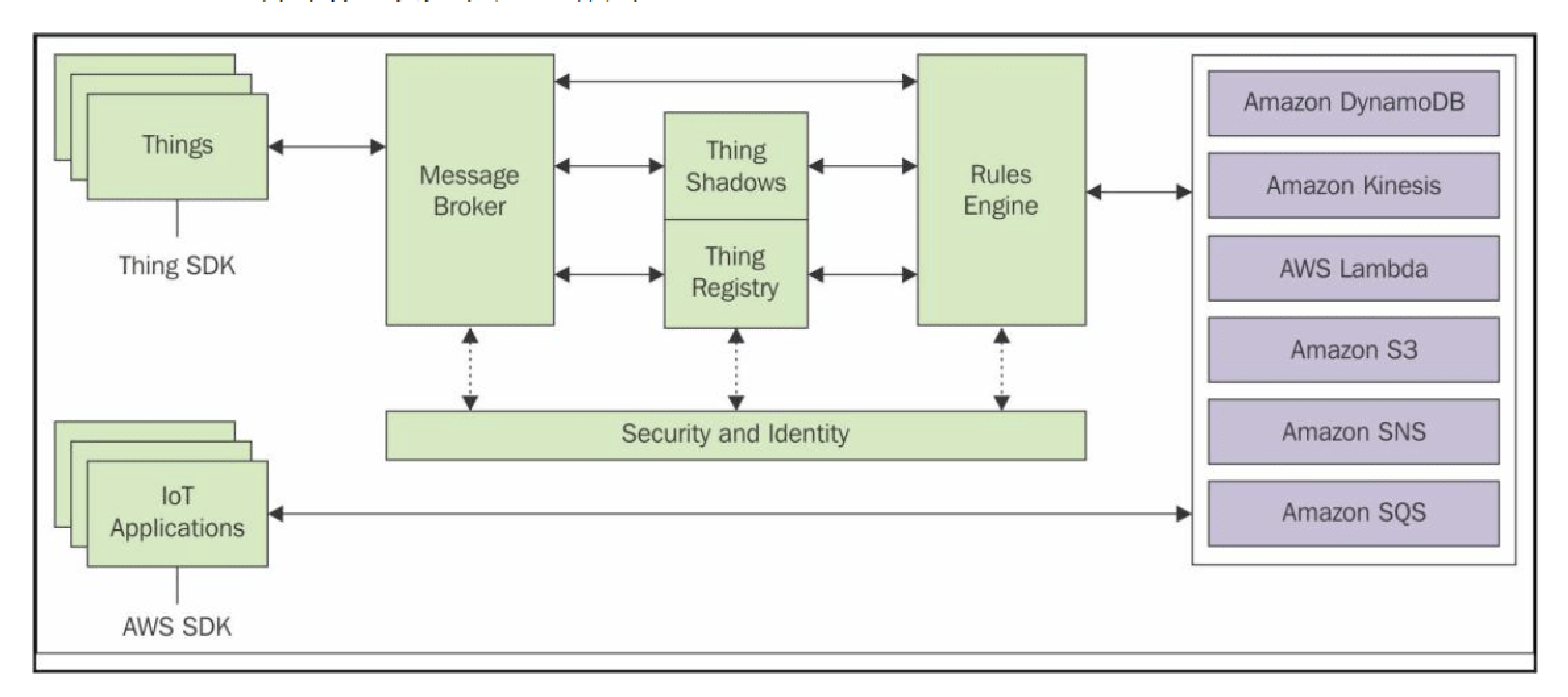
图 6-3

更多关于微软的信息，可以访问 https://azure.microsoft.com/。本书不会介绍所有的微软 Azure 服务，下面的部分将介绍 Azure IoT。

亚马逊 AWS IoT

亚马逊提供的云技术被称为亚马逊 AWS。现在亚马逊也提供 IoT 解决方案。IoT 板可以从 AWS 连接、推送或者获取数据。

AWS IoT 架构大致如图 6-4 所示。



来源：http://docs.aws.amazon.com/iot/latest/developerguide/aws-iot-how-it-works.html

图 6-4

可以通过 Message Broker 看到 IoT 板连接到 AWS IoT。可以应用一些规则来帮助过滤数据。在后端网站，亚马逊有许多服务来管理数据，如 Amazon DynamoDB、Kinesis、AWS Lambda 和 Amazon S3。这些有助于优化 IoT 进行数据处理。

更多关于 AWS IoT 的信息可以访问 http://docs.aws.amazon.com/iot/latest/developerguide/aws-iot-how-it-works.html。

Arduino 云

Arduino（arduino.cc）提供了一个免费的云平台供用户自己的 Arduino 连接。详情访问官方网站 https://cloud.arduino.cc/，如图 6-5 所示。需要先注册才能访问。

目前，Arduino 云提供 MQTT broker，可以从一个 Arduino 板发送消息到另一个 Arduino 板。

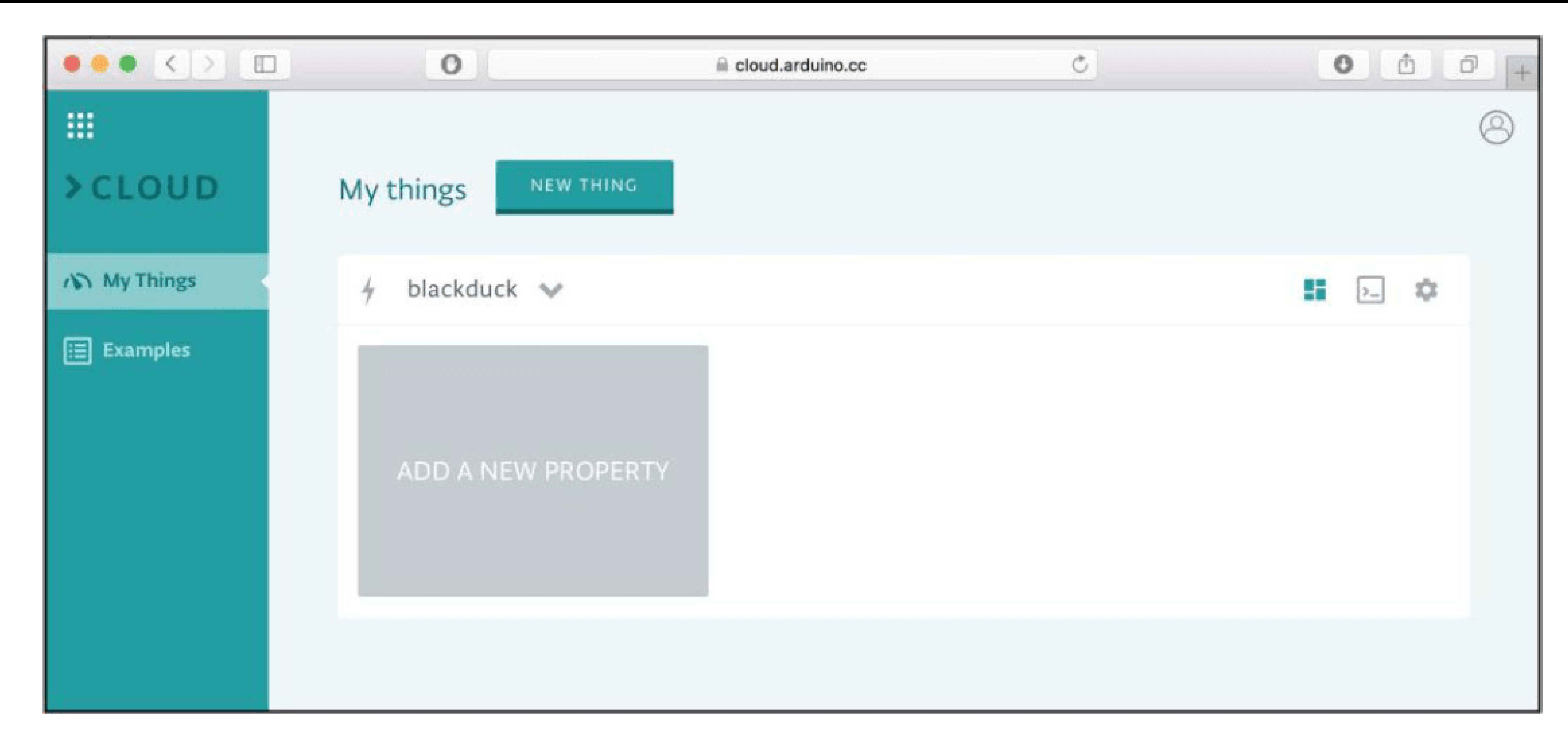


图 6-5

如果想要使用 Arduino 云，首先需要有一个特定的网络模块或者 Arduino 板。下面是一些适用 Arduino 云的网络模块和 Arduino 板：

- Arduino/Genuino Yún Shield
- Arduino/Genuino MKR1000
- WiFi Shield 101

在本例中，使用 Arduino MKR1000 板用于开发（https://www.arduino.cc/en/Main/ArduinoMKR1000）。

设置 Arduino 云

注册完 Arduino 云的网站，可以从 Arduino 板访问 Arduino 云。第一步是注册 Arduino 板。

可以访问下面的链接完成注册：https://cloud.arduino.cc/cloud/getting-started，或者可以访问网址 https://cloud.arduino.cc/cloud 注册一个新设备。然后单击 NEW THING 按钮，显示如图 6-6 所示界面。

输入 Arduino 板的名字，如 arduinobot。完成后单击 SAVE 按钮保存名字。

可以在如图 6-7 所示的面板上看到 Arduino 板的名字。

现在定义一些传感器属性。在场景里属性是温度和湿度，这两个属性将被 Arduino 板用来上传和下载传感器数据。

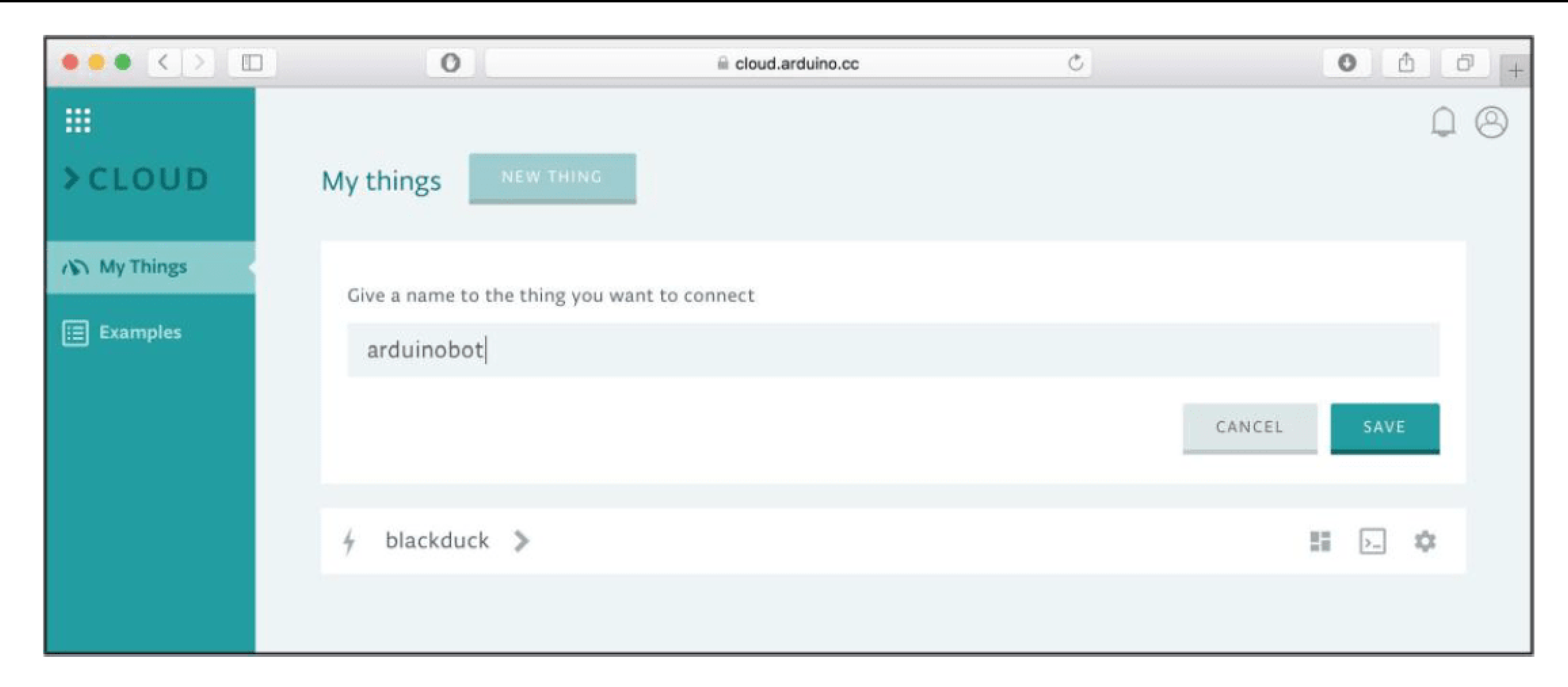


图 6-6

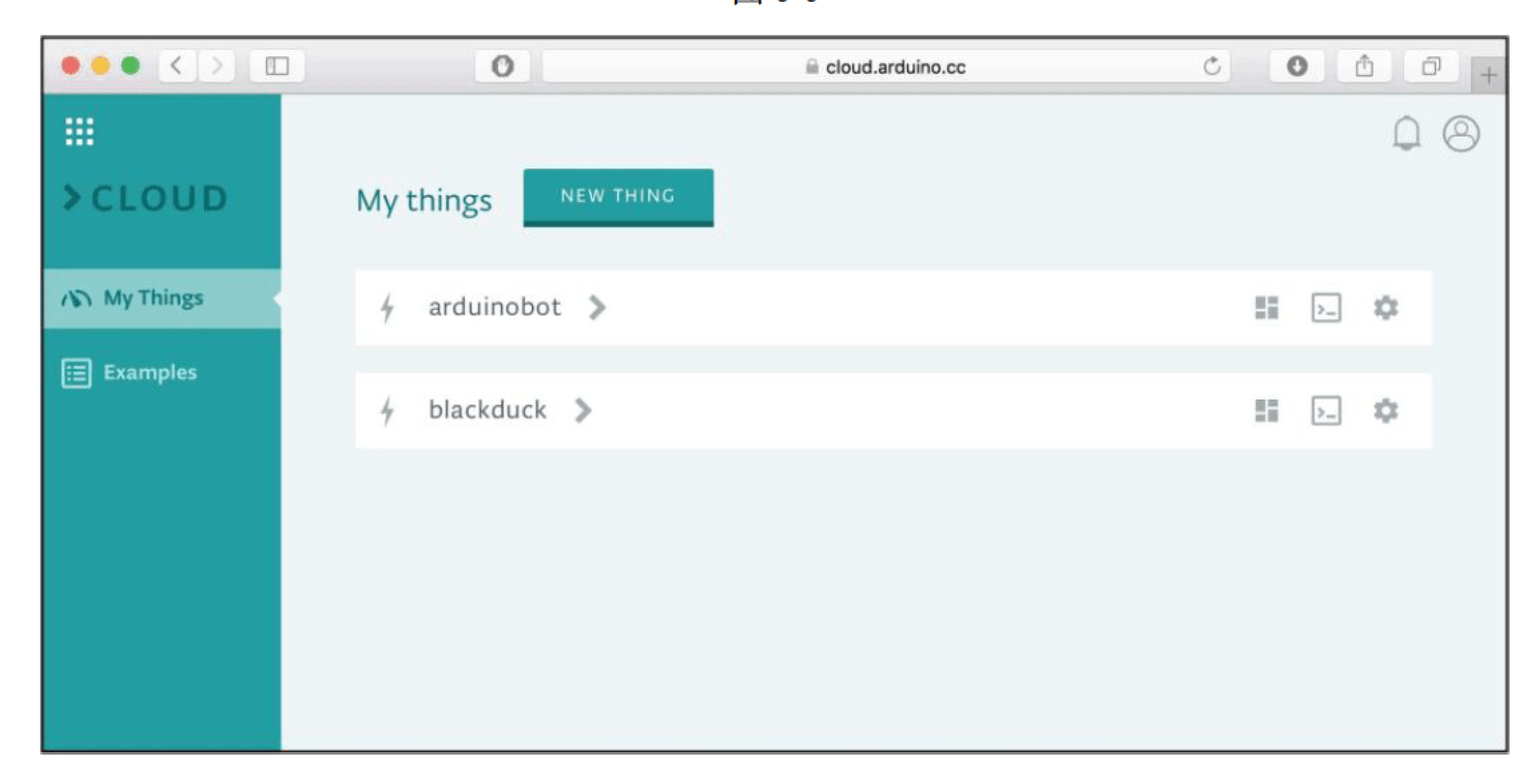


图 6-7

在注册好的板子上单击左侧的图标展示板子属性，可以看到如图 6-8 所示的表单。单击 ADD A NEW PROPERTY 按钮，然后输入以下值。

- Name：Humidity。
- Type：Float。
- Policy：Update when the value changes。

完成后单击 SAVE 按钮保存属性，如图 6-9 所示。

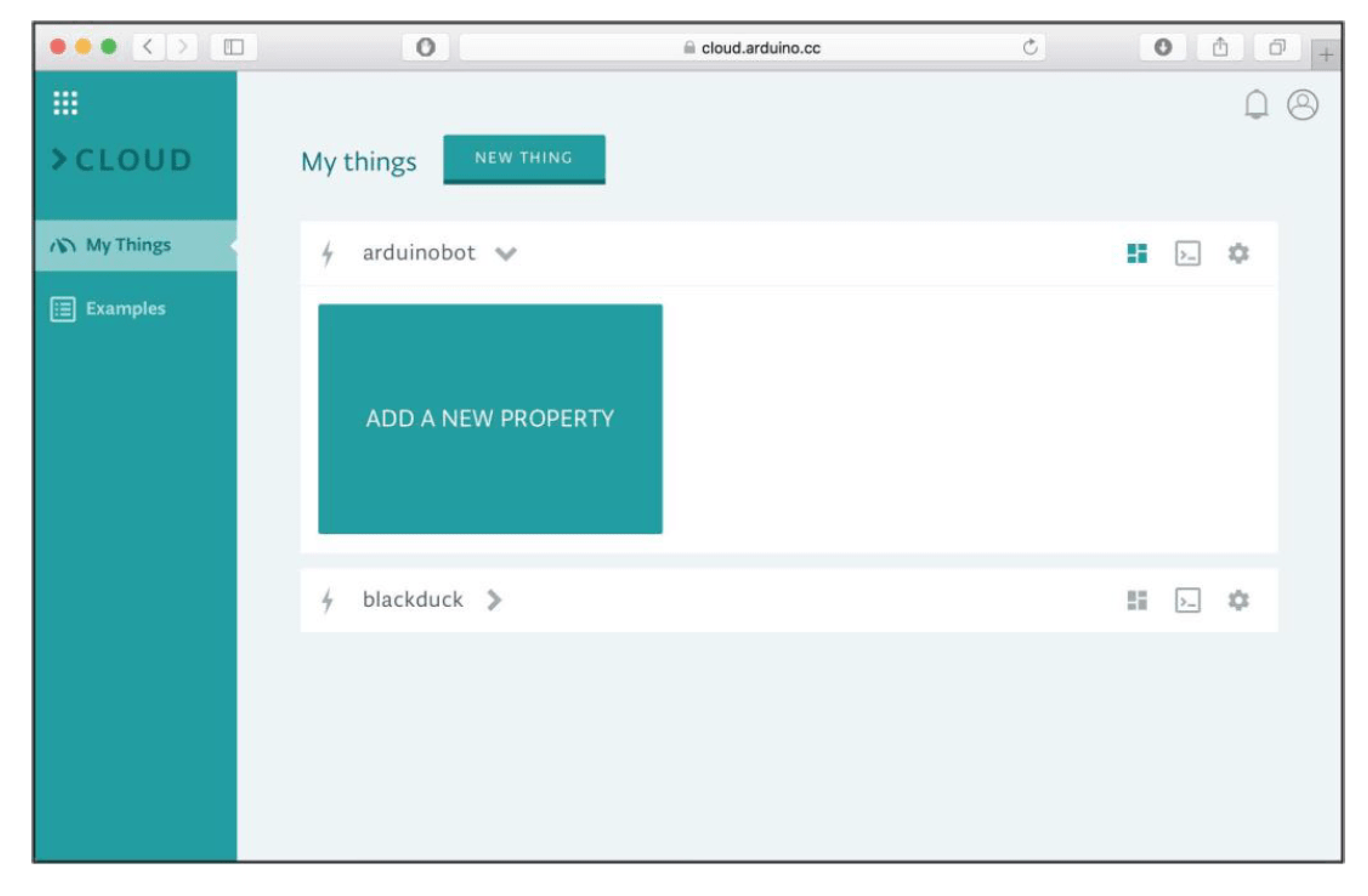

cloud.arduino.cc
CLOUD
My Things
Examples
My things
NEW THING
arduinobot
ADD A NEW PROPERTY
blackduck

图 6-8

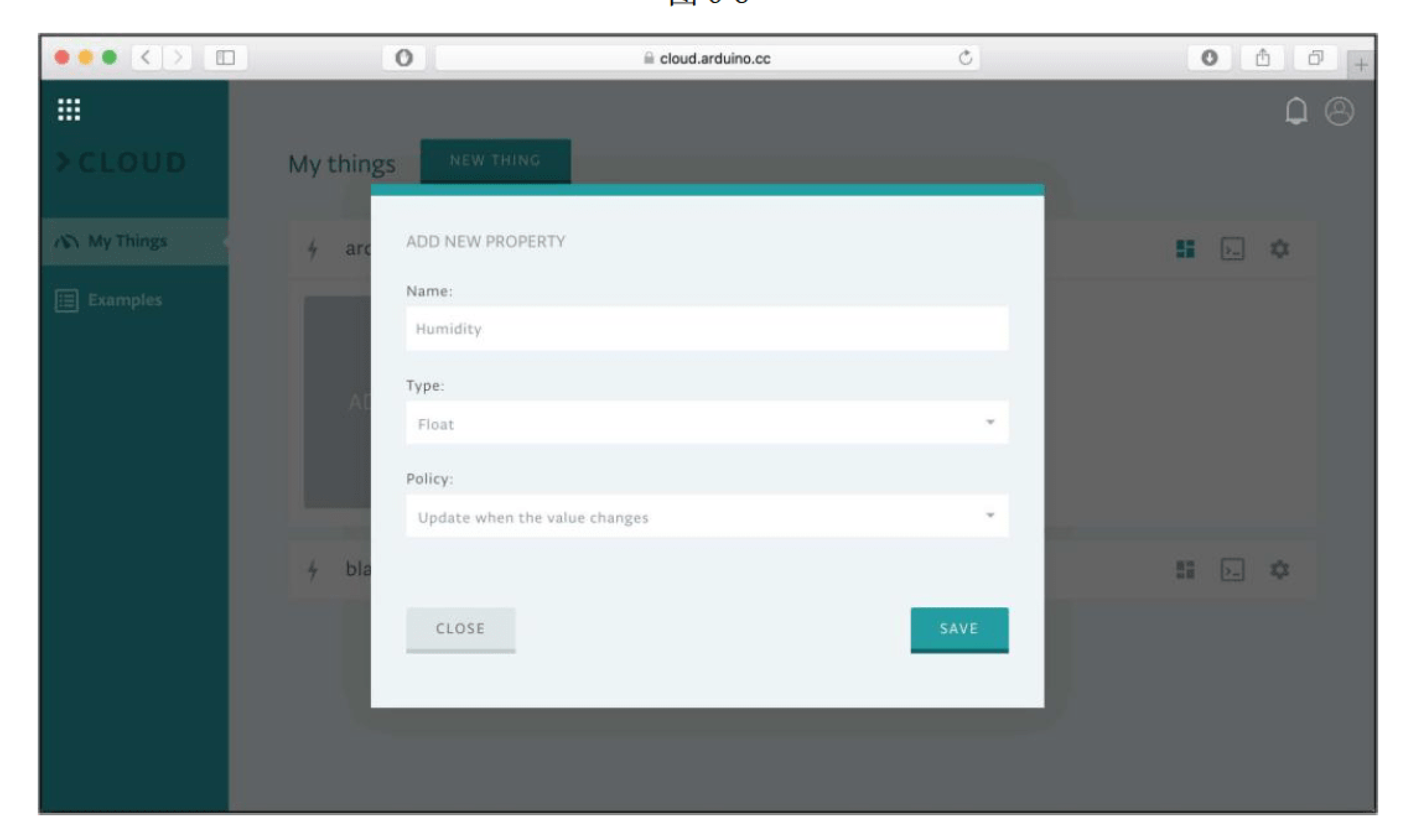

cloud.arduino.cc
CLOUD
My Things
Examples
My things
NEW THING
ADD NEW PROPERTY
Name:
Humidity
Type:
Float
Policy:
Update when the value changes
CLOSE
SAVE

图 6-9

用相同操作创建第二个属性。

- ❑ Name：Temperature。
- ❑ Type：Float。
- ❑ Policy：Update when the value changes。

现在可以看到这两个创建好的属性，如图 6-10 所示。

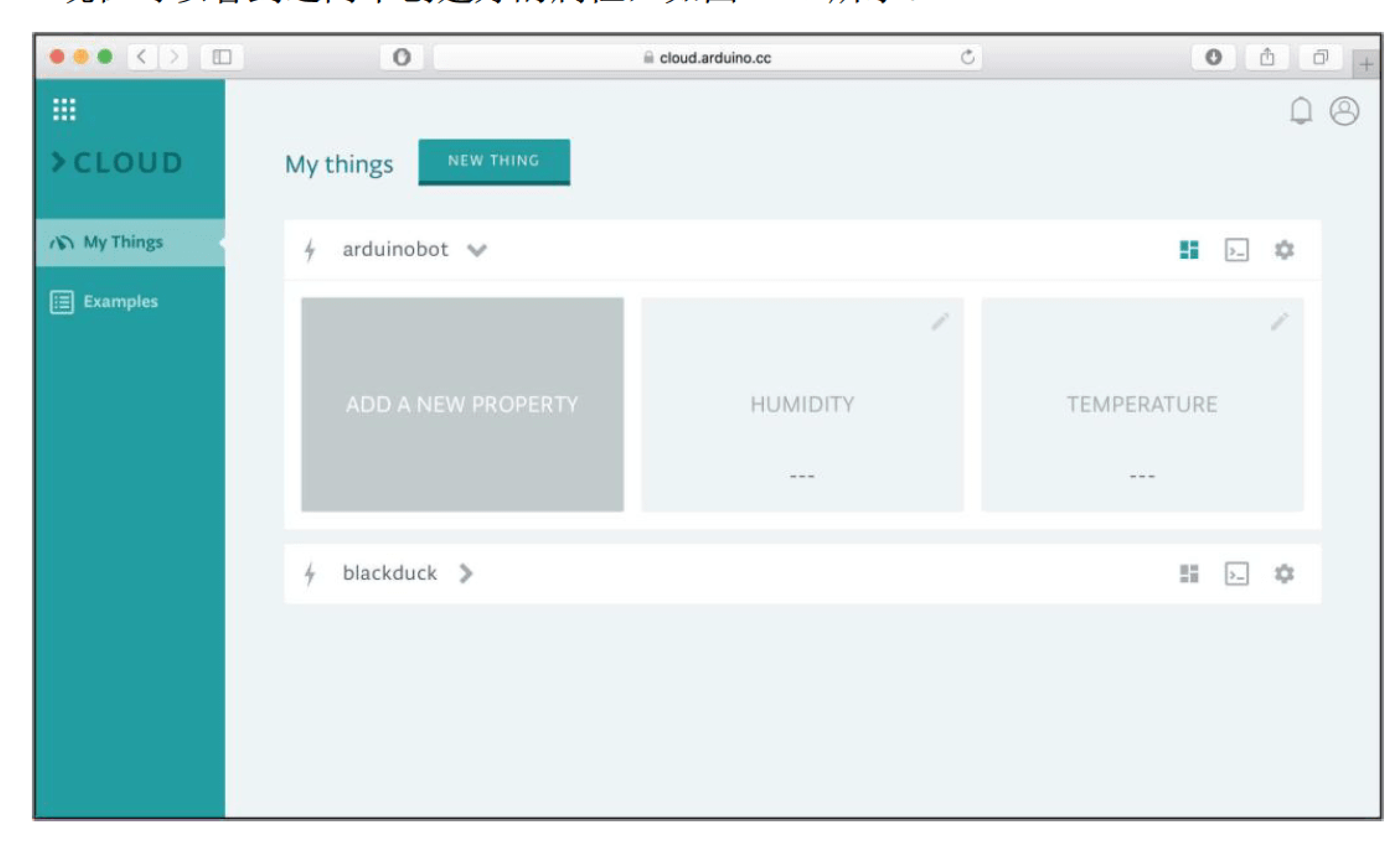


图 6-10

这两个 Arduino 板的属性将被用来作为数据目标。下面将展示如何访问这些属性。

布线

这里使用一个 DHT22 传感器来获取温度和湿度的数据。在前面已经学习了如何在板子上使用 DHT22。

在这个例子中，连接 DHT22 传感器模块到 Arduino MKR1000。连线如下：

- ❑ VDD（pin 1）连接到 Arduino VCC（3.3V）pin。
- ❑ SIG（pin 2）连接到 Arduino digital pin 8。
- ❑ GND（pin 4）连接到 Arduino GND。

可以看到连线如图 6-11 所示。

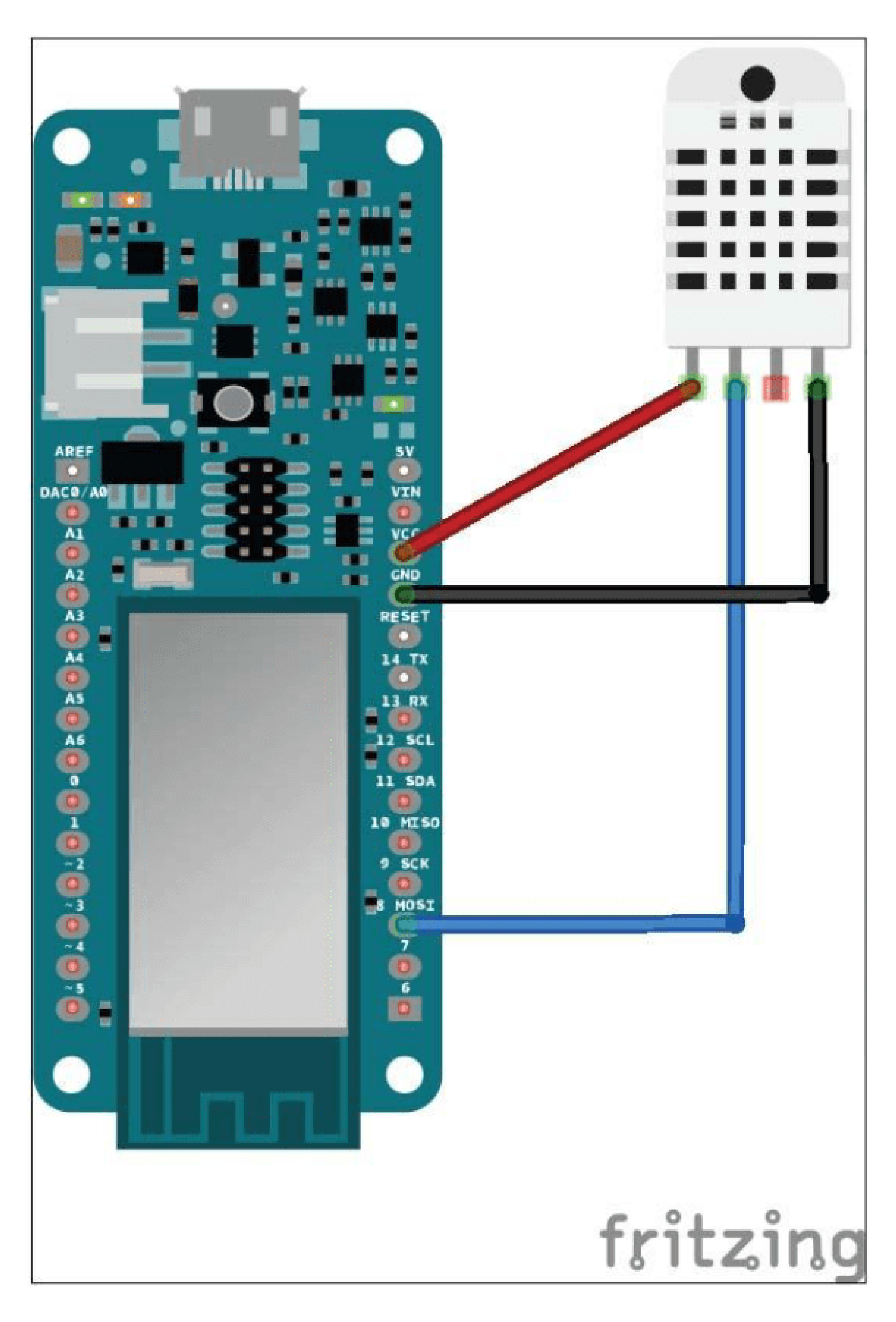


图 6-11

添加 Arduino 云库

Arduino 云提供 MQTT 协议下载和上传数据。在 Arduino 平台上，可以使用 ArduinoCloud 库来访问 Arduino 云服务器。这个库是开源的，可以在 https://github.com/arduino-libraries/ArduinoCloud 中看到所有代码。

为了安装 ArduinoCloud，使用 Arduino IDE。选择菜单 Sketch | Include Library | Manage Libraries。之后应该可以看到一个如图 6-12 所示的 Library Manager 对话框。

输入 arduinocloud，然后可以找到 ArduinoCloud 库。单击右侧的 Install 按钮。

安装完成后就可以开发 Sketch 程序。

升级 Arduino 云网站的 SSL 证书

如果用的是 Arduino MKR1000 或者是带有 WiFi101 shield 的 Arduino 板，需要为 Arduino 云网站（arduino.cc）升级 SSL 证书。

注意：下载最新的固件更新软件https://github.com/arduino-libraries/WiFi101-FirmwareUpdater/releases。

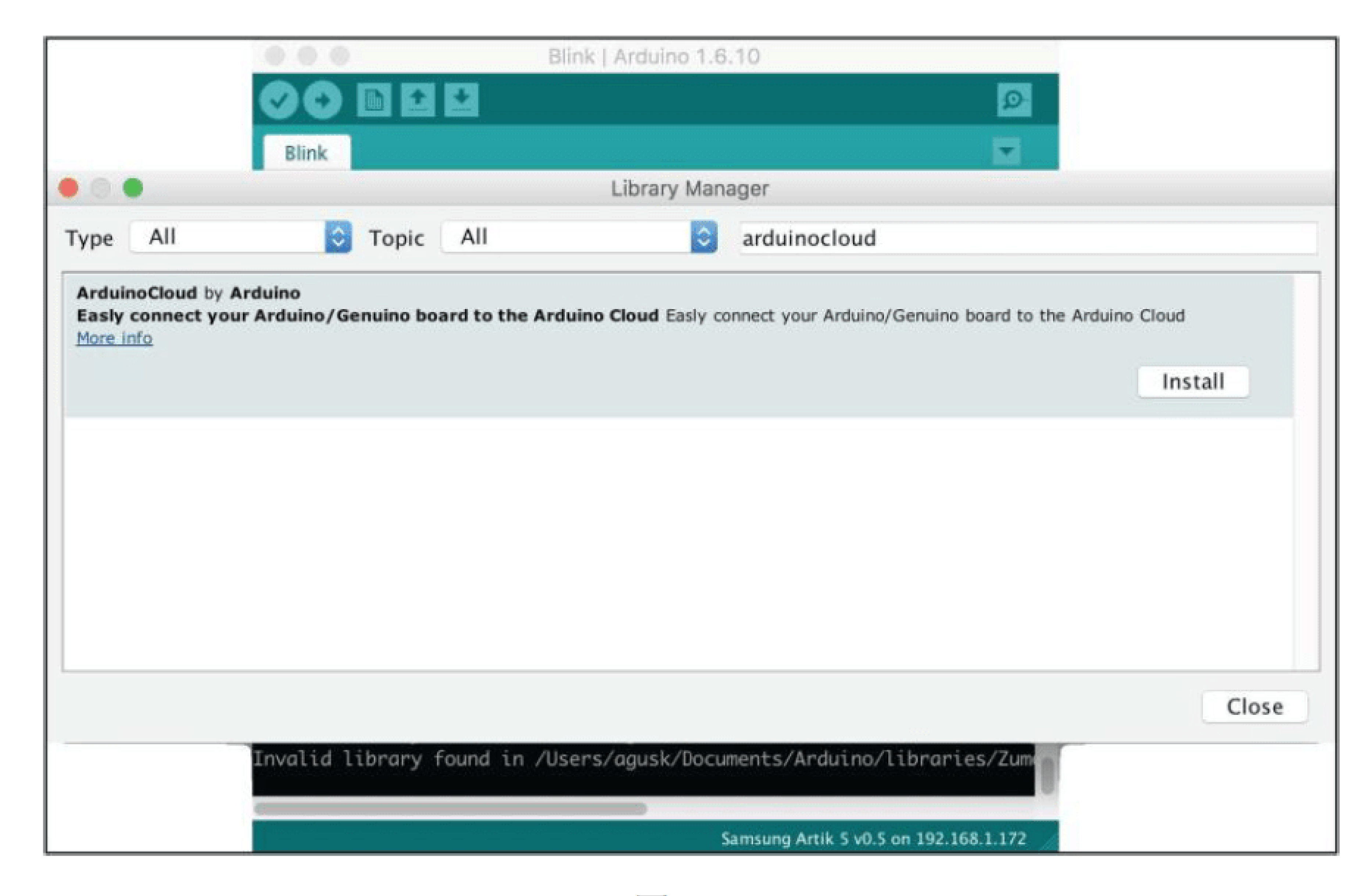


图 6-12

打开 Arduino IDE。选择菜单 File | Examples | WiFi101 | FirmwareUpdater，打开 FirmwareUpdater 程序，看到如图 6-13 所示的 Sketch 程序。



图 6-13

在 Arduino IDE 里选择 Arduino 板的目标和端口，然后编译并上传到 Arduino 板。

也可以选择在 Arduino 板连接的计算机上运行 WINC1500 SSL 证书更新软件。输入如下命令：

```
$ ./winc1500-uploader-gui
```

可以看到如图 6-14 所示的 WINC1500 SSL 证书更新软件。

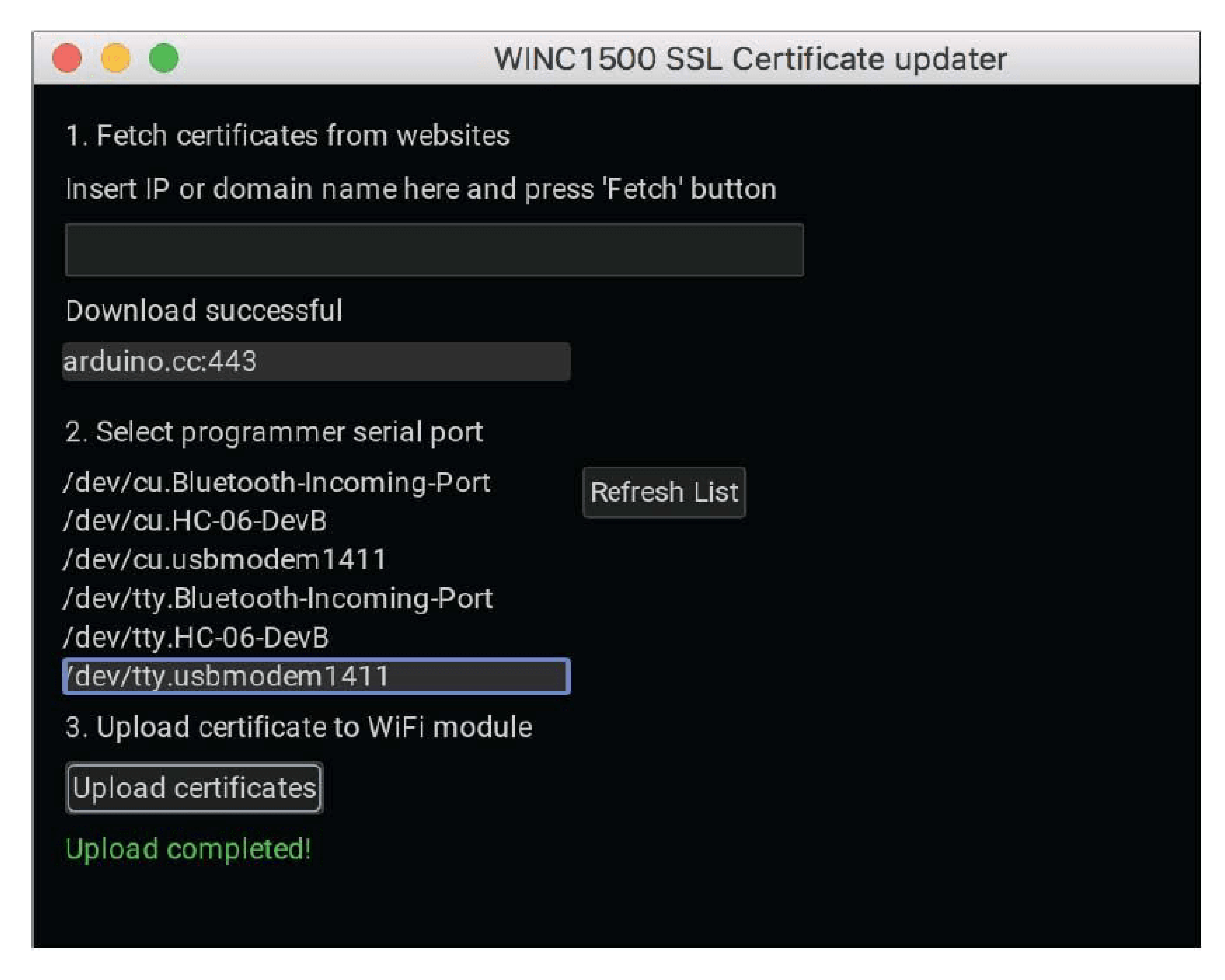


图 6-14

接着执行以下 4 个步骤：

（1）在第一个文本框里输入 arduino.cc（Fetch certificates from websites）。

（2）单击 Fetch 按钮，从 arduino.cc 下载 SSL 证书。

（3）在选项 2 里选择一个 Arduino 端口。

（4）完成后单击 Update certificates 按钮上传 SSL 证书到 Arduino 板。

这样 WINC1500 固件已经在 Arduino.cc 网站上有更新过的 SSL 证书。

为 Arduino 云编写程序

接着为该例子编写 Sketch 程序。使用 Arduino 云上注册过的 Arduino 板的示例代码作为基础。单击中间的图标，可以看到如图 6-15 所示的 Sketch 代码。

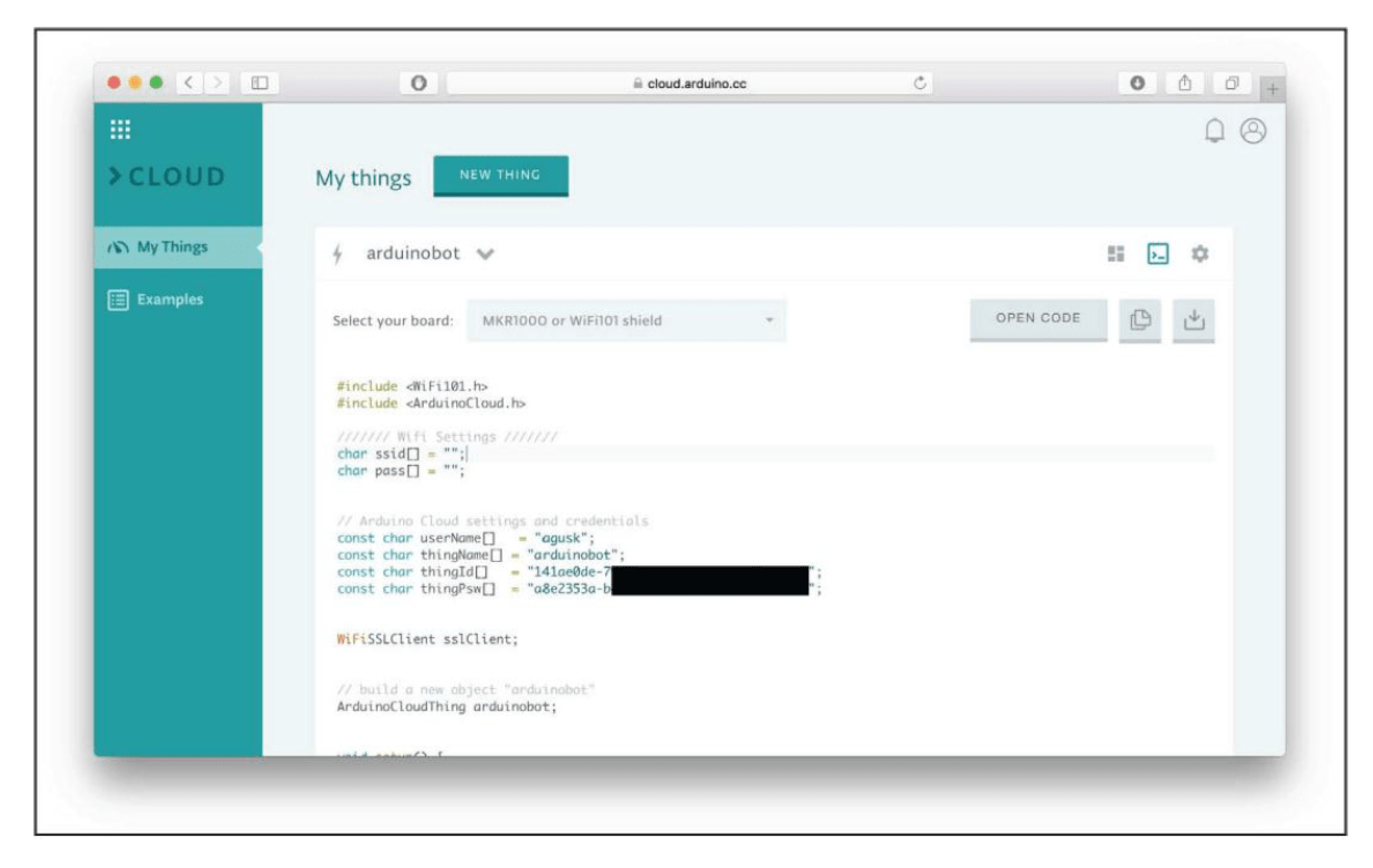


图 6-15

复制并把 Sketch 程序粘贴到 Arduino IDE。

下一步修改 Sketch 代码来集成 DHT22 传感器。程序通过 DHT22 读取温度和湿度值，然后发送传感器数据到 Arduino 云。

编写如下完整代码：

```
#include <WiFi101.h>
#include <ArduinoCloud.h>
#include "DHT.h"

/////// Wi-Fi 设置 ///////
char ssid[] = "<your ssid>";
char pass[] = "<your ssid password";


//Arduino 云设置和证书
const char userName[]   = "<your_thing_username>";
const char thingName[] = "<your_thins_name>";
const char thingId[]   = "<your_thing_id>";
const char thingPsw[]  = "<your_thing_password>";
```

```
WiFiSSLClient sslClient;


//创建新对象"arduinobot"
ArduinoCloudThing arduinobot;


//定义 DHT22
#define DHTTYPE DHT22
//在 DHT22 定义 pin
#define DHTPIN 8

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin (9600);

  dht.begin();

  //试图连接 Wi-Fi
  Serial.print("Attempting to connect to WPA SSID: ");
  Serial.println(ssid);

  while (WiFi.begin(ssid, pass) != WL CONNECTED) {
    //失败 4s 后重试
    Serial.print("failed ... ");
    delay(4000);
    Serial.print("retrying ... ");
  }
  Serial.println("connected to wifi");

  arduinobot.begin(thingName, userName, thingId, thingPsw, sslClient);
  arduinobot.enableDebug();

  //定义属性
  arduinobot.addProperty("Humidity", FLOAT, R);
  arduinobot.addProperty("Temperature", FLOAT, R);
}

void loop() {
```

```
  arduinobot.poll();

  delay(2000);

  //读取温度和湿度需要 250ms
  //超过 2s 的传感器都有点“老了”
  float h = dht.readHumidity();
  //默认温度读取单位是摄氏度
  float t = dht.readTemperature();


  //检查是否读取失败并及时退出
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  arduinobot.writeProperty("Temperature", t);
  arduinobot.writeProperty("Humidity", h);
  delay(1000);
}
```

修改代码里 Wi-Fi 和 Arduino 云的值，完成后保存为 ch06_01。

现在编译并上传程序到 Arduino 板。

从 Arduino IDE 打开串口监视器工具。可以看到程序连接到 Wi-Fi 并且得到来自 DHT22 的温度和湿度值。读取完传感器数据之后，程序将其发送到 Arduino 云。

如图 6-16 所示是在本例串口监视器上的程序输出。

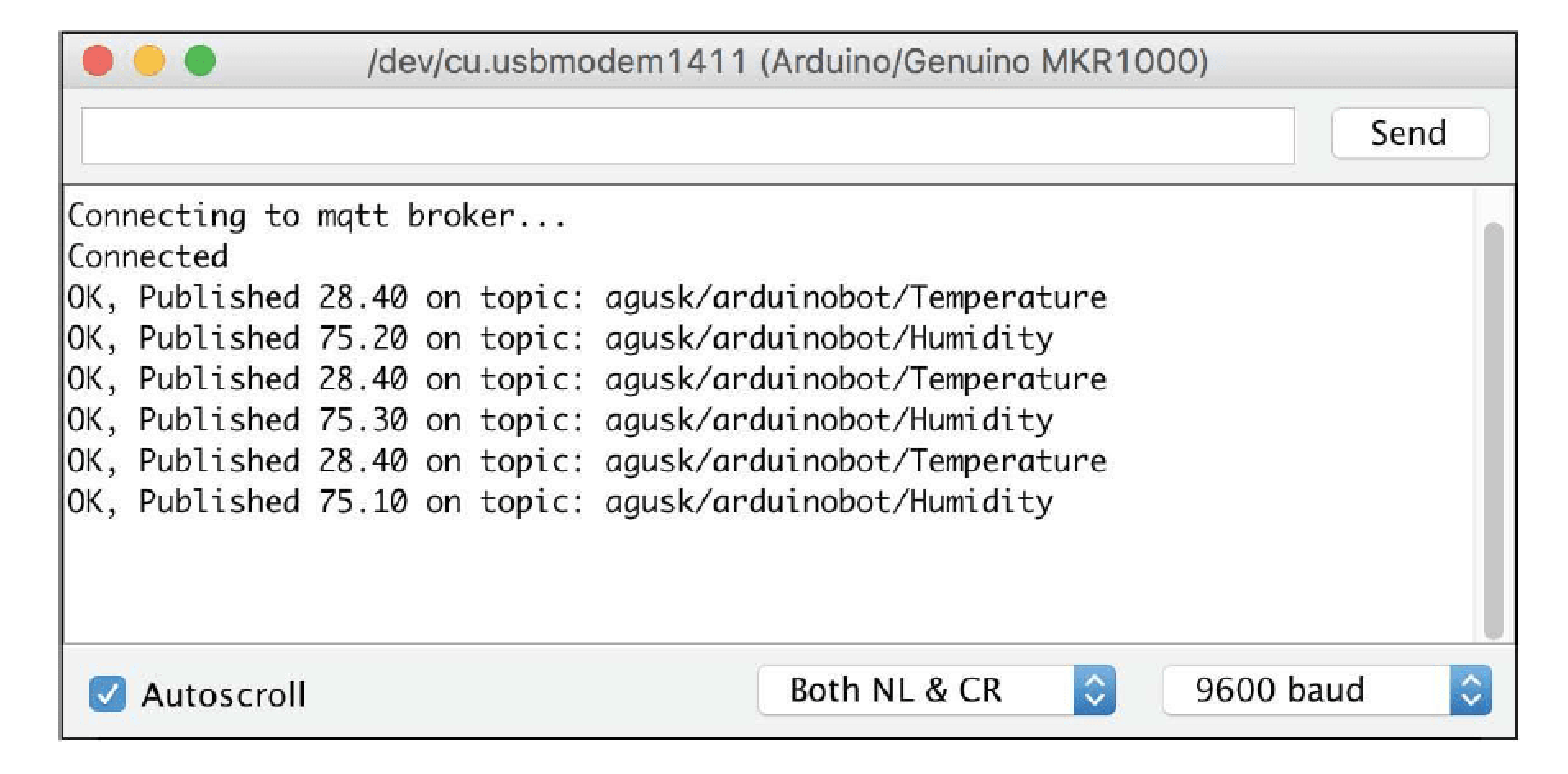


图 6-16

打开 Arduino 云网站，然后转到注册的 Arduino 板，可以看到温度和湿度值，如图 6-17 所示。

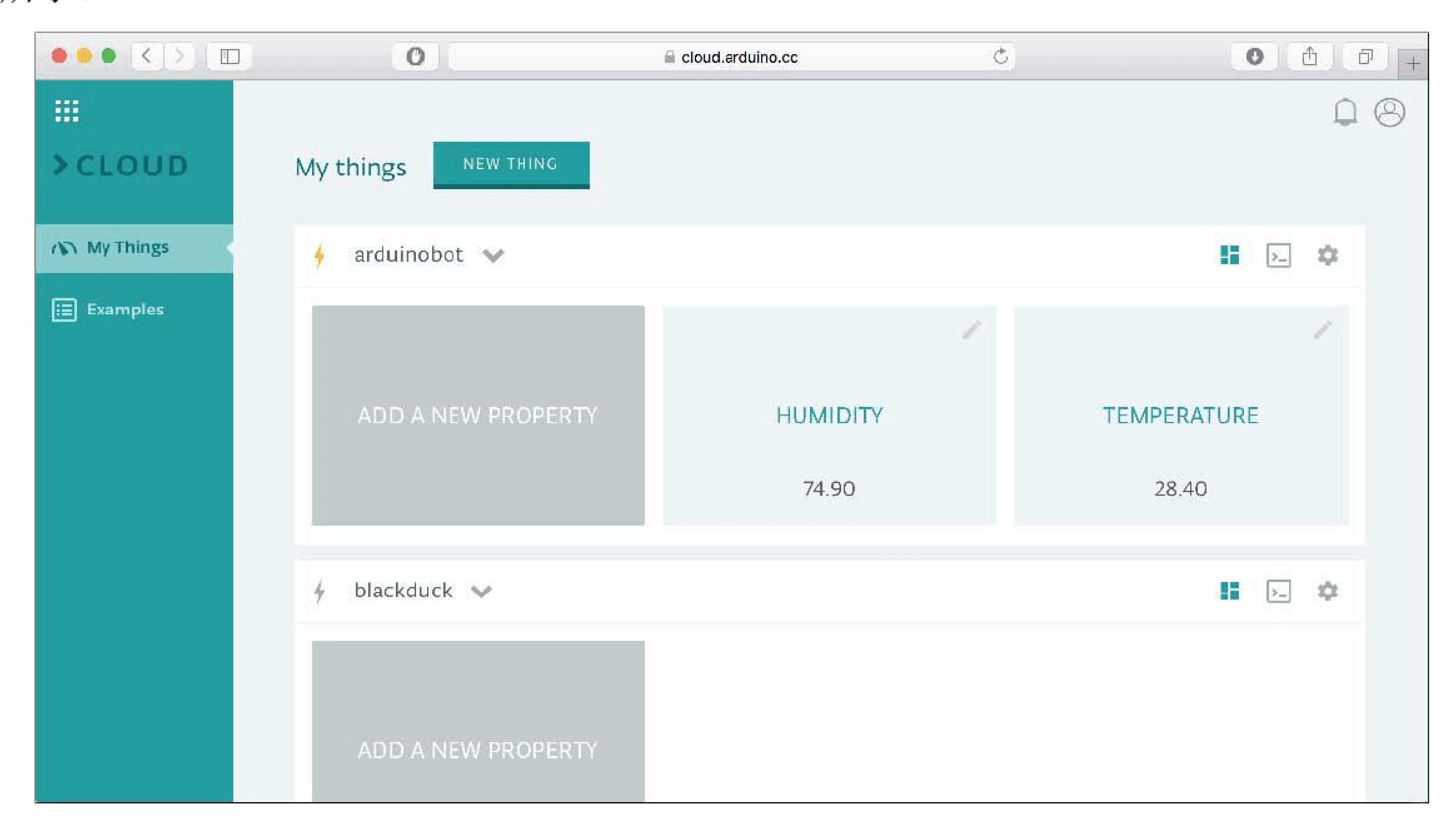


图 6-17

那么它是如何工作的呢？代码其实很简单。在 setup()函数里激活串口和模块并试图连接 Wi-Fi：

```
Serial.begin (9600);
dht.begin();


while (WiFi.begin(ssid, pass) != WL CONNECTED) {
    //失败 4s 后重试
    Serial.print("failed ... ");
    delay(4000);
    Serial.print("retrying ... ");
}
```

接着调用 ArduinoCloud 库并定义 Arduino 板属性：

```
arduinobot.begin(thingName, userName, thingId, thingPsw, sslClient);
arduinobot.enableDebug();
```

```
//定义属性
arduinobot.addProperty("Humidity", FLOAT, R);
```

在 loop()函数里向 Arduino 云发起 poll()请求：

```
arduinobot.poll();
delay(2000);
```

下一步是通过 DHT22 传感器读取温度和湿度值，然后发送到 Arduino 云：

```
float h = dht.readHumidity();
float t = dht.readTemperature();
if (isnan(h) || isnan(t)) {
  Serial.println("Failed to read from DHT sensor!");
  return;
}

arduinobot.writeProperty("Temperature", t);
arduinobot.writeProperty("Humidity", h);
delay(1000);
```

使用微软 Azure IoT Hub

Azure IoT Hub 提供可靠的 device-to-cloud 和 cloud-to-device 的超大规模消息。每个设备都有安全证书的访问控制来保证通信安全，也为流行的编程语言和平台提供了设备库。

很多种类的 IoT 板都可以连接到 Azure IoT Hub，可以在 https://azure.microsoft.com/en-us/develop/iot/get-started/检查是否可以连接。

在本节，会尝试用 Raspberry Pi 或者笔记本连接微软 Azure IoT Hub，也可以使用其他板子。

设置微软 Azure IoT Hub

设置微软 Azure IoT Hub 之前需要在 Azure 有一个激活的会员账户，当然微软也提供试用账户。

打开浏览器并转到 https://portal.azure.com/打开微软 Azure。在左侧的菜单找 IoT Hub，然后可以看到如图 6-18 所示的 IoT Hub dashboard。

在 IoT Hub dashboard 上单击+Add 添加一个新的 IoT hub，然后可以看到如图 6-19 所

示的表单。

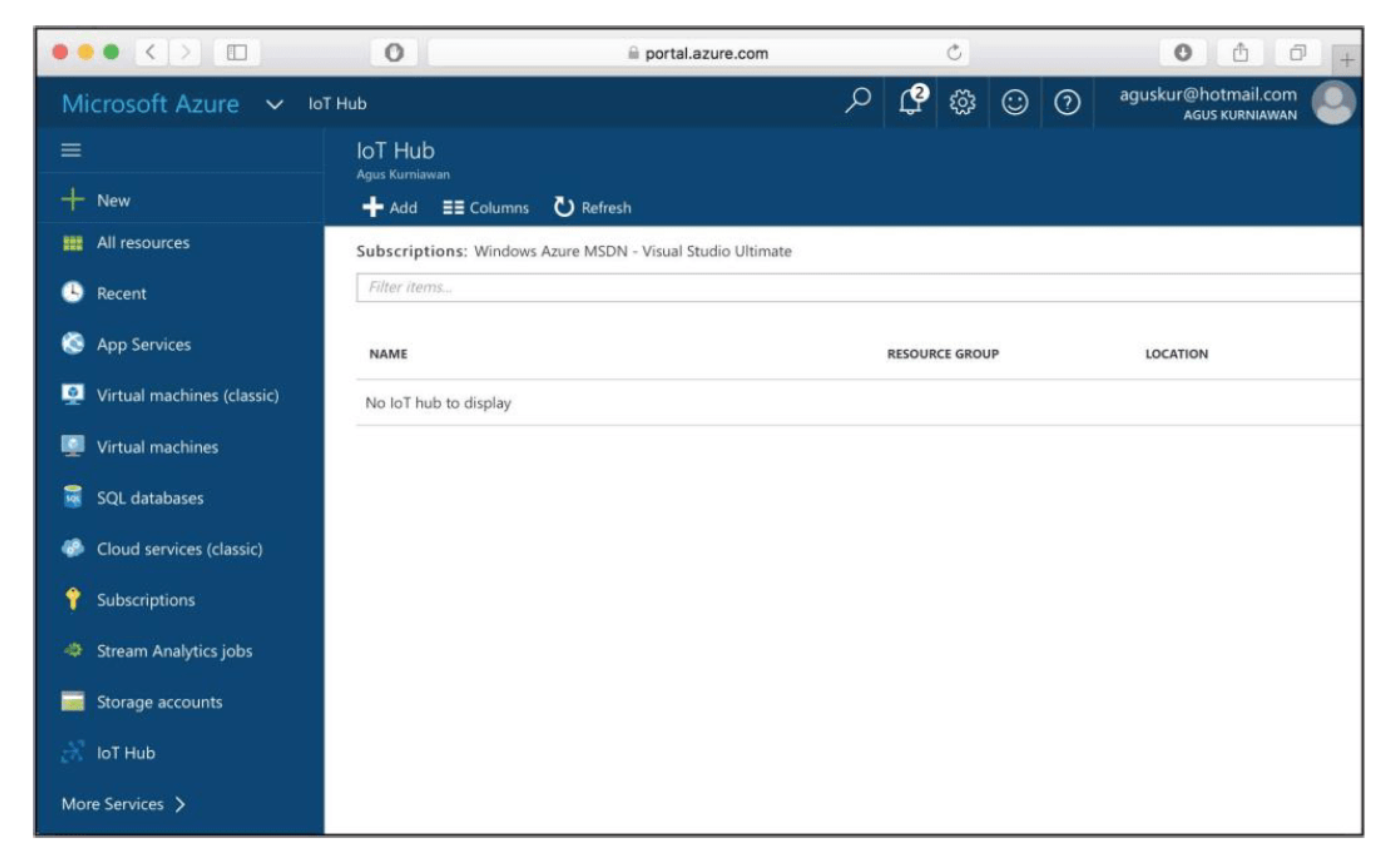


图 6-18

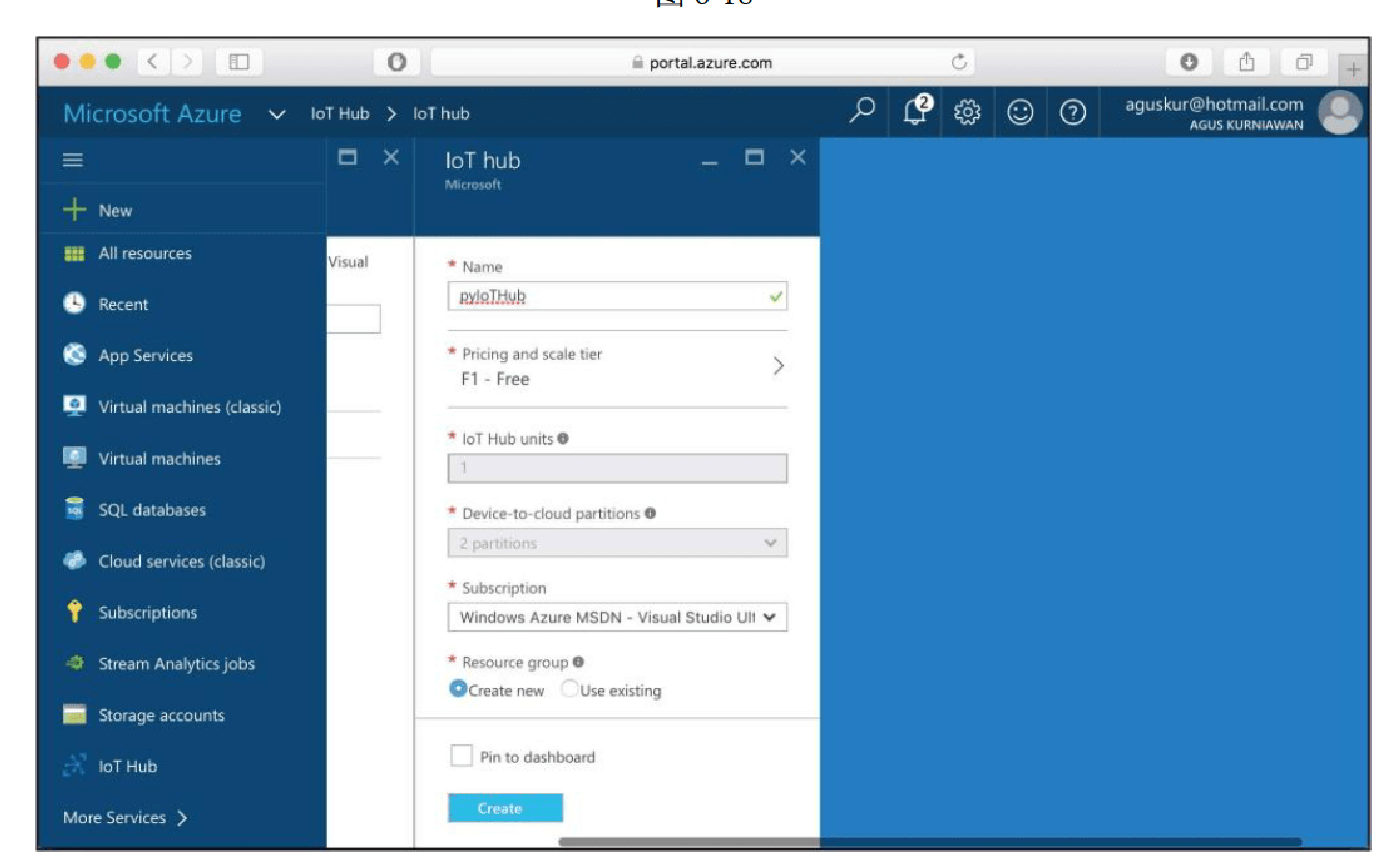


图 6-19

填好需要的字段，包括价格和规模。填写时，也有免费选项，为 F1。

完成后单击 Create 按钮，显示如图 6-20 所示的界面。

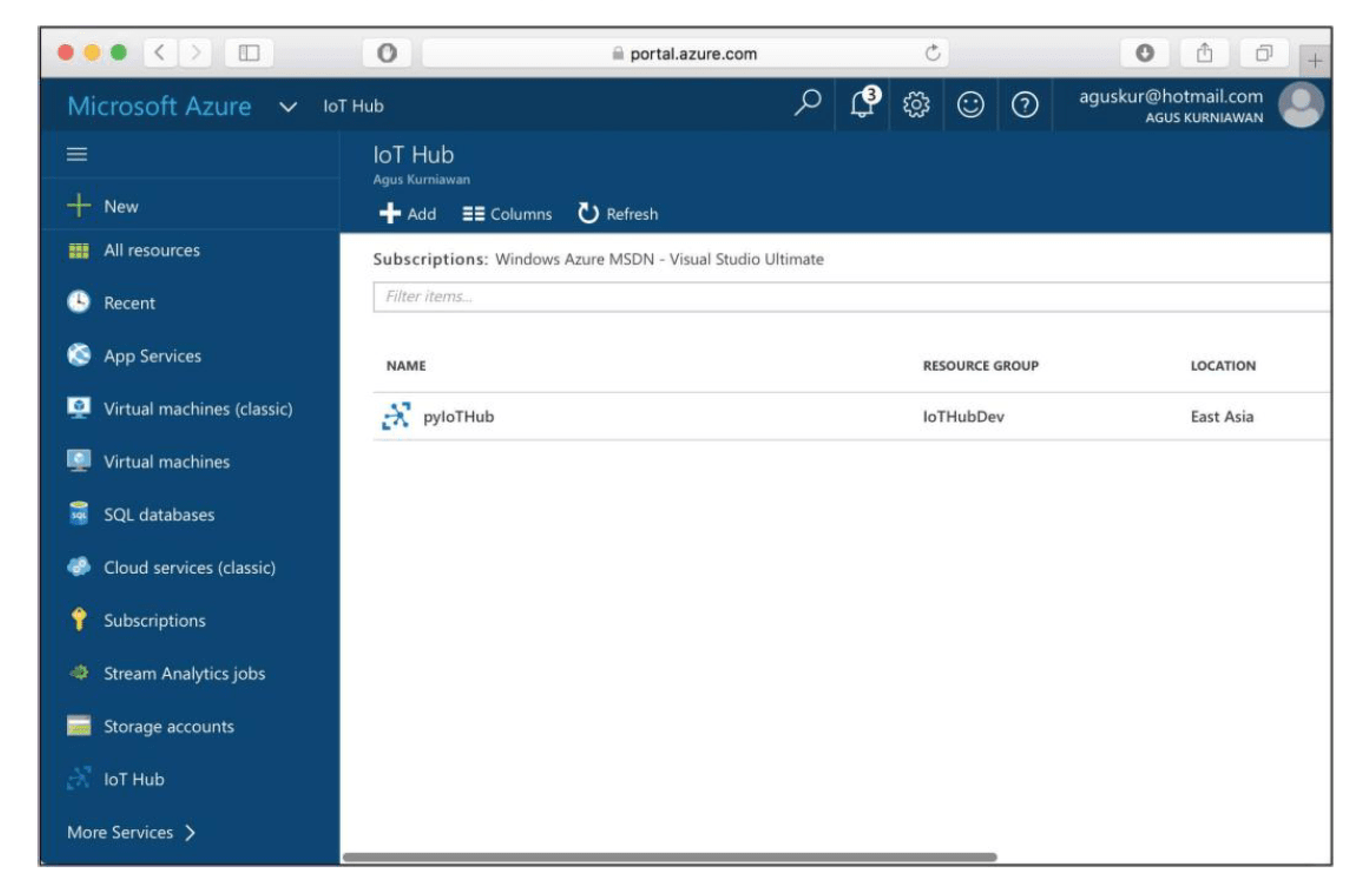


图 6-20

微软 Azure IoT Hub 将创建新的 hub。完成需要等待一会。

新的 Azure IoT hub 创建完毕后，就可以进行个性化设置。

注册 IoT 设备

为了让 IoT 板使用微软 IoT Hub，需要先注册 IoT 板，这样才可以获得访问证书。

有两种方法可以注册 IoT 设备。

打开 Microsoft Azure | IoT Hub | Settings，可以看到如图 6-21 所示的设置。

单击 Shared 按钮可以看到如图 6-22 所示的一些访问策略。

单击 iothubowner 策略，可以看到共享的访问密钥。复制“连接字符串”的值——primary key。

“连接字符串”用于添加一个新的 IoT 板访问 Azure IoT。目前可以用两种办法：iothub-explorer 或者设备探测工具。

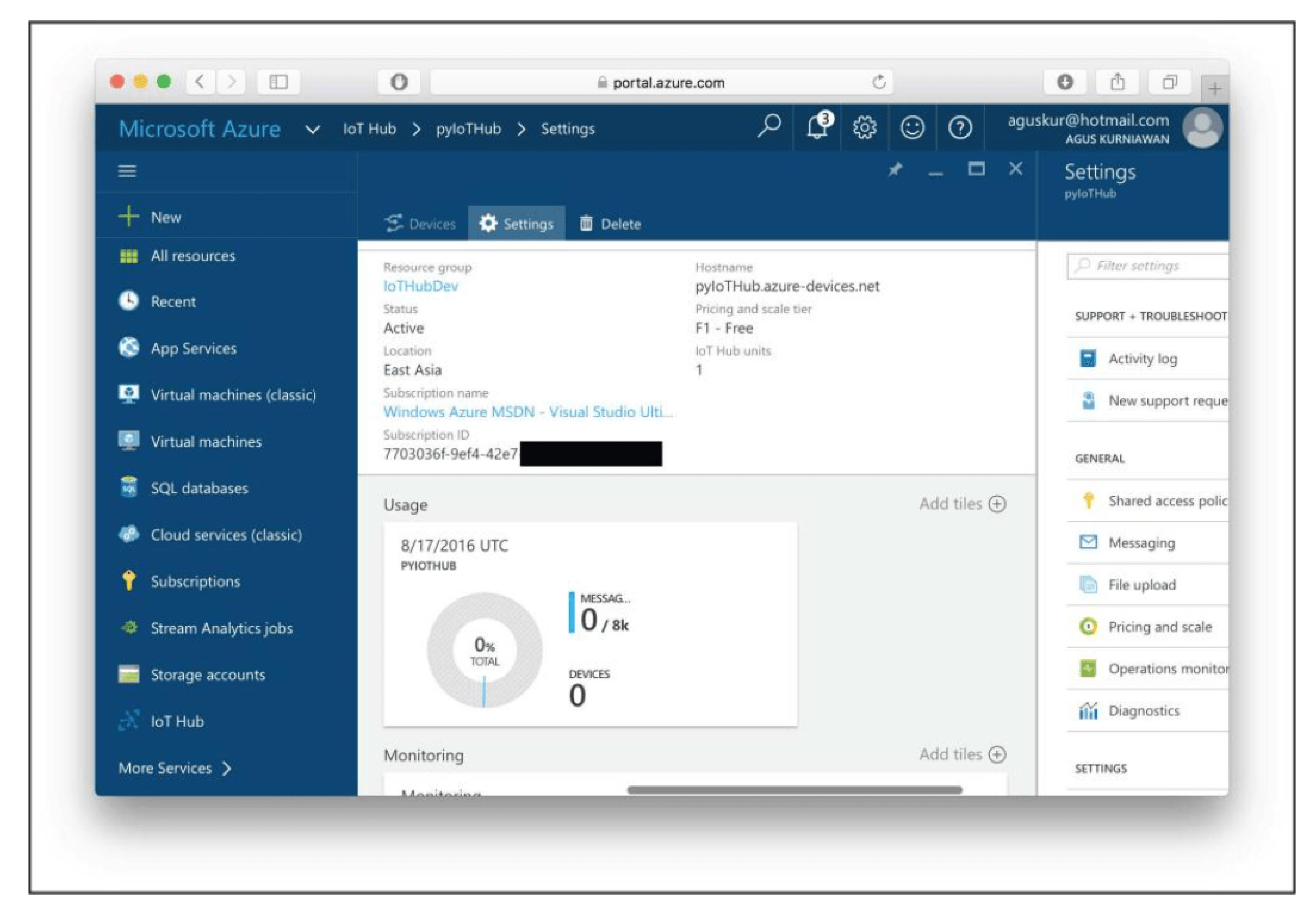

portal.azure.com
Microsoft Azure
IoT Hub > pyloTHub > Settings
aguskur@hotmail.com
AGUS KURNIAWAN
New
All resources
Recent
App Services
Virtual machines (classic)
Virtual machines
SQL databases
Cloud services (classic)
Subscriptions
Stream Analytics jobs
Storage accounts
IoT Hub
More Services >
Devices
Settings
Delete
Resource group
IoTHubDev
Status
Active
Location
East Asia
Subscription name
Windows Azure MSDN - Visual Studio Ulti...
Subscription ID
7703036f-9ef4-42e7[redacted]
Hostname
pyloTHub.azure-devices.net
Pricing and scale tier
F1 - Free
IoT Hub units
1
Usage
Add tiles
8/17/2016 UTC
PYIOTHUB
0% TOTAL
MESSAG...
0 / 8k
DEVICES
0
Monitoring
Settings
pyloTHub
Filter settings
SUPPORT + TROUBLESHOOT
Activity log
New support reque
GENERAL
Shared access polic
Messaging
File upload
Pricing and scale
Operations monitor
Diagnostics
SETTINGS

图 6-21

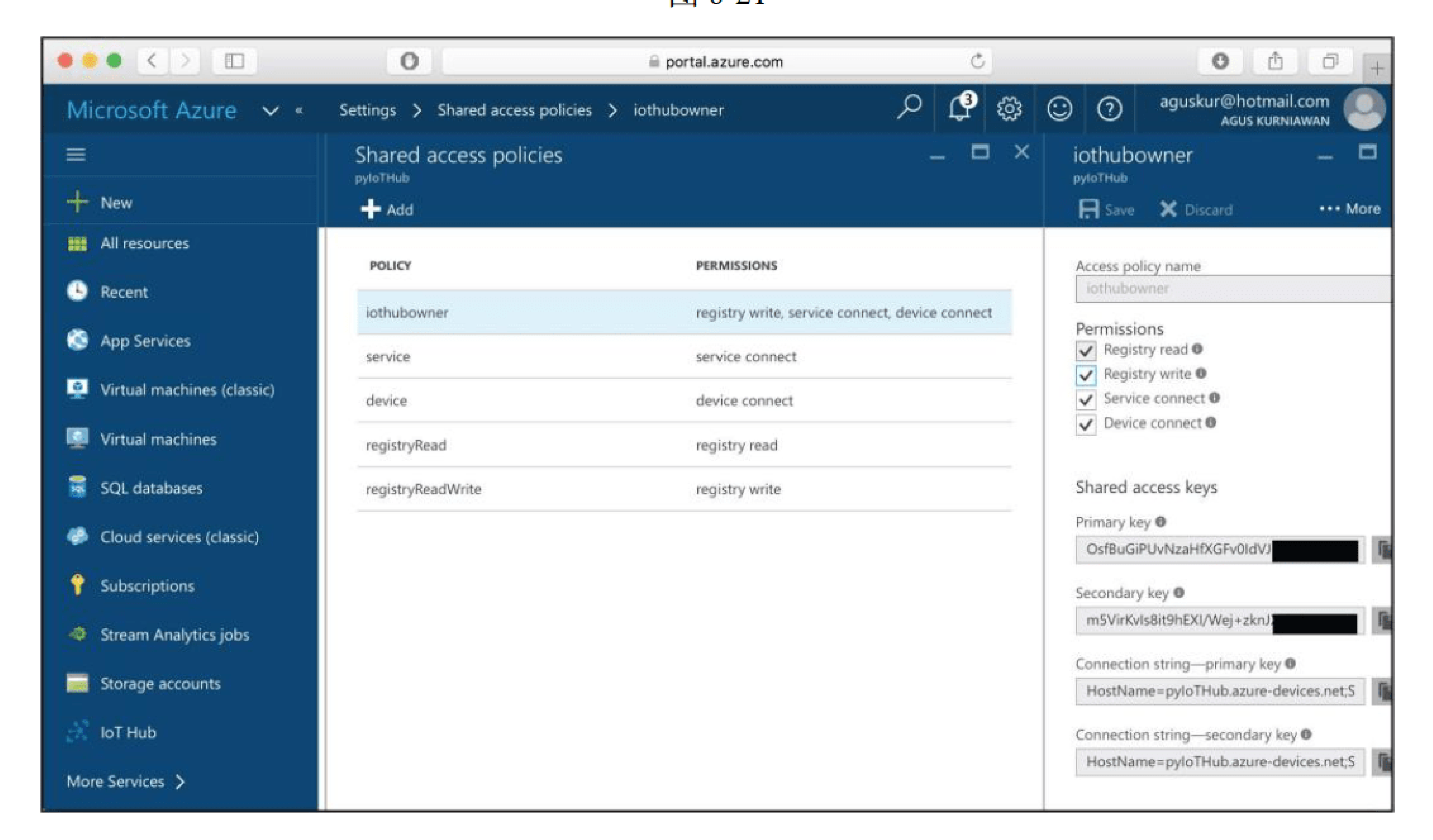

portal.azure.com
Microsoft Azure
Settings > Shared access policies > iothubowner
aguskur@hotmail.com
AGUS KURNIAWAN
New
All resources
Recent
App Services
Virtual machines (classic)
Virtual machines
SQL databases
Cloud services (classic)
Subscriptions
Stream Analytics jobs
Storage accounts
IoT Hub
More Services >
Shared access policies
pyloTHub
Add
POLICY
PERMISSIONS
iothubowner
registry write, service connect, device connect
service
service connect
device
device connect
registryRead
registry read
registryReadWrite
registry write
iothubowner
pyloTHub
Save
Discard
More
Access policy name
iothubowner
Permissions
Registry read
Registry write
Service connect
Device connect
Shared access keys
Primary key
OsfBuGiPUvNzaHfXGFv0ldVJ[redacted]
Secondary key
m5VirKvls8it9hEXI/Wej+zknJ[redacted]
Connection string—primary key
HostName=pyloTHub.azure-devices.net;S
Connection string—secondary key
HostName=pyloTHub.azure-devices.net;S

图 6-22

iothub-explorer 工具是一个基于 Node.js 的命令行工具，可以运行在任意平台。设备探测工具是在 Windows 平台的有图形界面的工具，不过平台仅限 Windows 系统。

推荐读一篇文章，网址为 https://github.com/Azure/azure-iot-sdks/blob/master/tools/iothub-explorer/readme.md，以了解更多如何使用 iothub-explorer 工具的信息。

本节将分享如何在 Windows 10 上使用设备探测工具注册一个新的 IoT 板。

可以从 https://github.com/Azure/azure-iot-sdks/releases 下载并安装设备探测工具。完成后，在配置一栏里的 Connection Information 文本框里复制“连接字符串”，如图 6-23 所示。

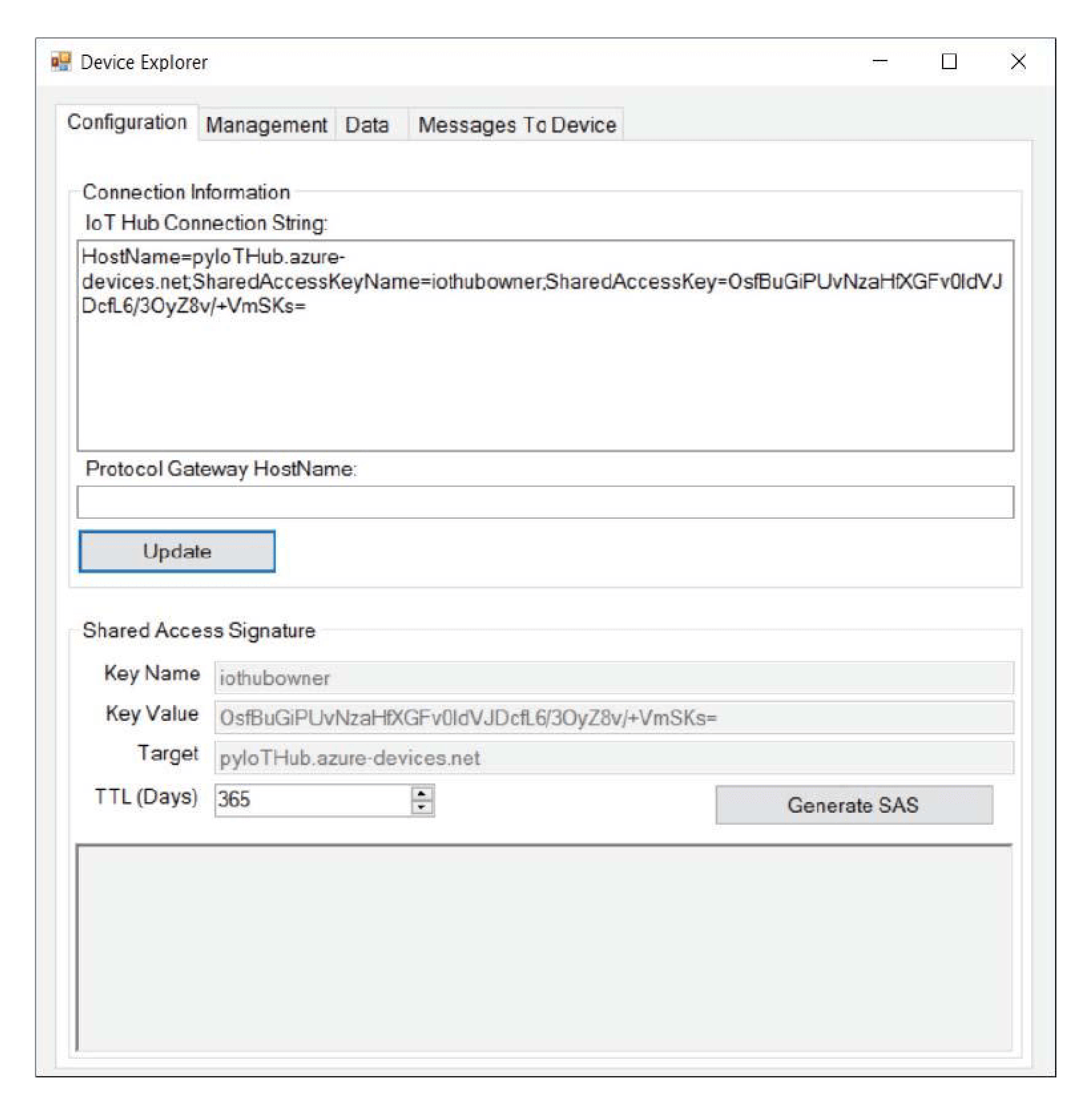


图 6-23

粘贴完 Azure IoT Hub 的“连接字符串”，单击 Update 按钮。如果看到 Management 一栏和图 6-24 所示的一样，就代表配置成功了。

单击 Create 按钮添加一个新的 IoT 设备，可以看到如图 6-25 所示的对话框。

输入设备的名字，然后单击 Create 按钮，在设备探测工具的 Management 一栏里能看到设备就代表成功了。

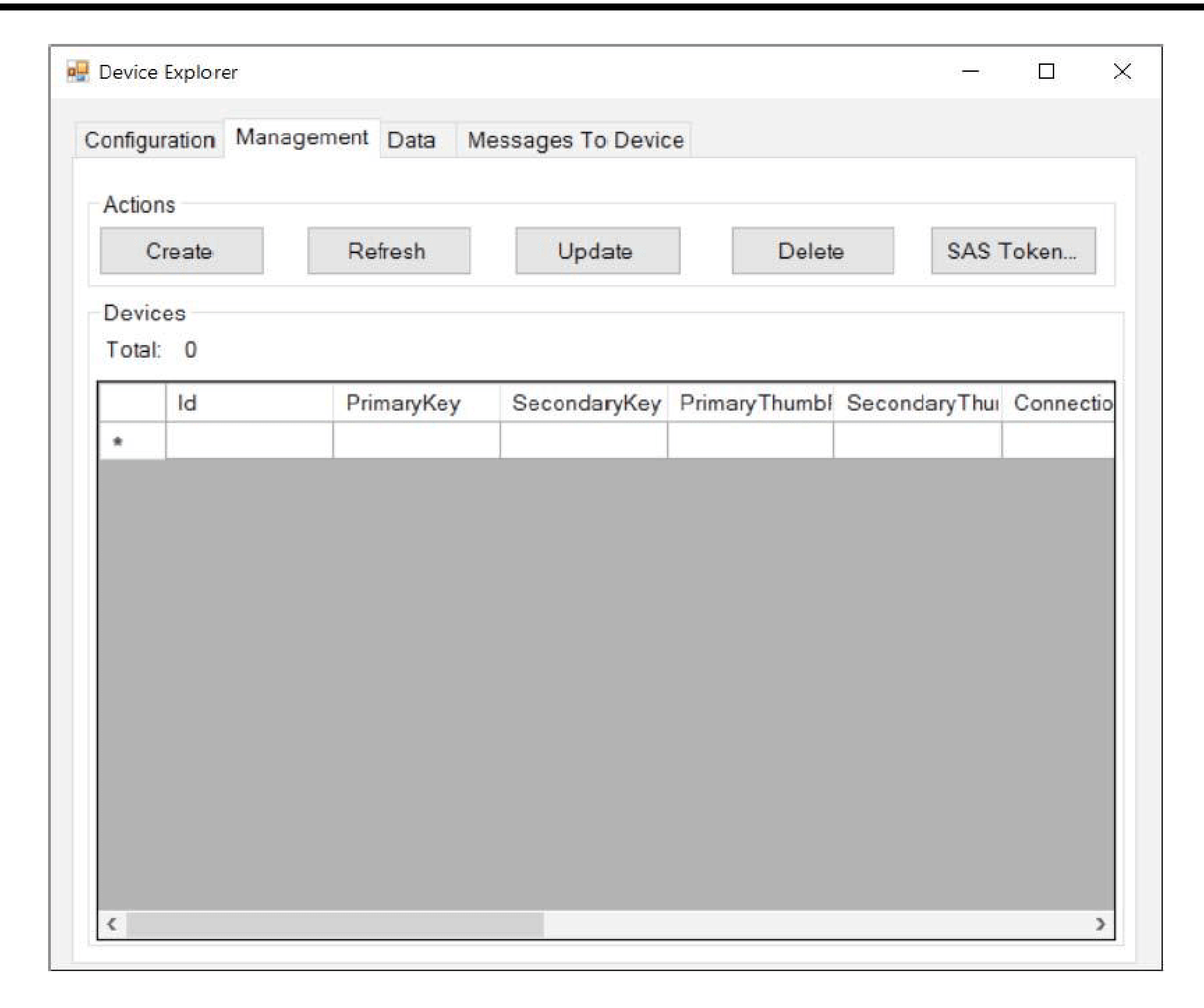


图 6-24

Create Device
Device Authentication
Security Keys
X509
Device ID: myraspi
Primary Key: WGatLwySoZRWAsRJqMtB33qpVoFrDAwicXTYSomvfR4=
Secondary Key: c9DKKmyLx4YfsD7MXBWY9Vkif9BhSNEfxuB7+ztzlqM=
Auto Generate ID
Auto Generate Keys
Create
Cancel

图 6-25

接着可以从 IoT 设备得到“连接字符串”。右击设备可以看到一个内容菜单，如图 6-26 所示。

选择 Copy connection string for selected device，然后将其粘贴到任意一个文本里，这个值将在后面的程序里用到。

下一节将进行解释。

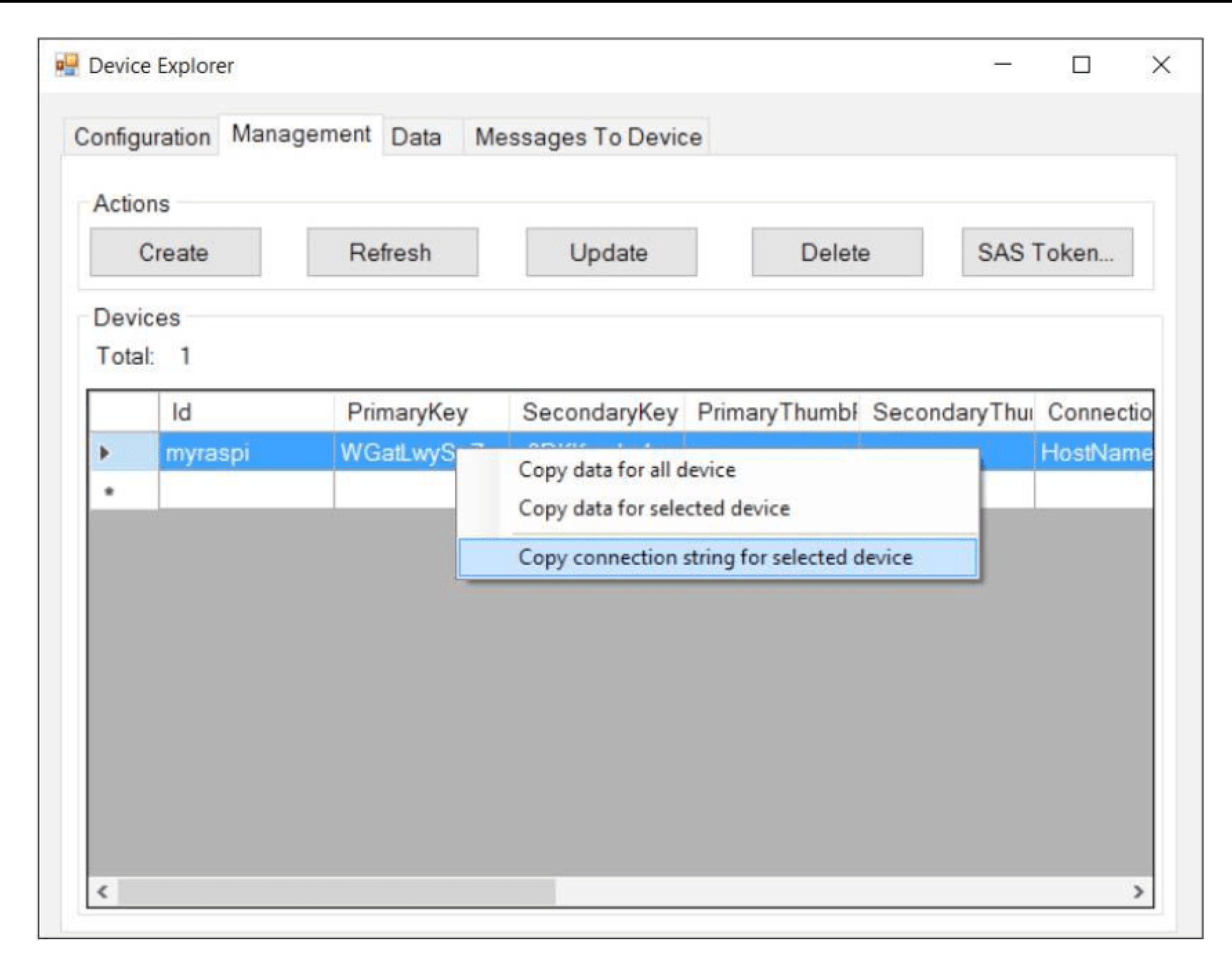


图 6-26

编写程序

下面将使用 Python 来为 Azure IoT Hub 编写程序，这里使用带有 Raspbian OS 的 Raspberry Pi 作为测试。

可以从 https://github.com/Azure/azure-iot-sdks 下载并安装 Azure IoT SDK。安装之前可以提高一下 Raspberry Pi 的交换文件大小的设置。

打开命令行，输入如下命令：

```
$ sudo nano /etc/dphys-swapfile
```

找到如下行：

```
CONF_SWAPSIZE=100
```

修改为：

```
CONF_SWAPSIZE=1024
```

完成后重新启动交换文件服务或者重启 Raspberry Pi。

接着从源码 https://github.com/Azure/azure-iot-sdks.git 安装 SDK for Azure IoT。先安装 Azure IoT SDK for C，再安装 Python 库。

在 Raspberry Pi 终端输入下面的命令：

```
$ sudo apt-get update
$ git clone --recursive https://github.com/Azure/azure-iot-sdks.git
$ cd azure-iot-sdks/c/build_all/linux/
$ ./setup.sh
$ ./build.sh
```

接着安装 Python 库。转到/python/build_all/linux/并输入如下命令：

```
$ cd ../../../
$ cd python/build_all/linux/
$ ./setup.sh
$ ./build.sh
```

安装完成后，可以在<sdk_azure_iot_hub>/python/device/samples/文件下找到库文件。复制这个文件到项目路径下，或者将其放到 Raspberry Pi Python 库的路径里。

在此处的例子里，准备编写程序访问 Azure IoT。这里已经修改了 https://github.com/Azure/azure-iot-sdks 的 Python 示例代码。在这个例子里，将发送温度和湿度数据到 Azure IoT Hub。

编写如下脚本：

```
#!/usr/bin/env python

import random
import time
import sys
import iothub_client
from iothub_client import *


# messageTimeout - 消息过期的最长时间
message_timeout = 10000

receive_context = 0
avg_temperature = 0
avg humidity = 0
message count = 3
received count = 0

# global counters
receive callbacks = 0
send_callbacks = 0
```

```
# MQTT 作为传输协议
protocol = IoTHubTransportProvider.MQTT

#  包含 Hostname、Device Id 和 Device Key 格式的字符串
# "HostName=<host_name>;DeviceId=<device_id>;SharedAccessKey=<device_
key>"
connection_string = "[device connection string]"

msg_txt  =  "{\"deviceId\":  \"<device_id>\",\"temperature\":  %.2f,
\"humidity\": %.2f}"

# 一些嵌入式平台需要验证信息
def set certificates(iotHubClient):
    from iothub client cert import certificates
    try:
        iotHubClient.set option("TrustedCerts", certificates)
        print("set option TrustedCerts successful")
    except IoTHubClientError as e:
        print("set_option TrustedCerts failed (%s)" % e)


def receive_message_callback(message, counter):
    global receive_callbacks
    buffer = message.get_bytearray()
    size = len(buffer)
    print("Received Message [%d]:" % counter)
    print("Data: <<<%s>>> & Size=%d" % (buffer[:size].decode('utf-8'),
size))
    map properties = message.properties()
    key value pair = map properties.get internals()
    print("Properties: %s" % key value pair)
    counter += 1
    receive_callbacks += 1
    print("Total calls received: %d" % receive_callbacks)
    return IoTHubMessageDispositionResult.ACCEPTED


def send_confirmation_callback(message, result, user_context):
    global send_callbacks
    print(
        "Confirmation[%d] received for message with result = %s" %
        (user_context, result))
```

```
    map_properties = message.properties()
    print("message_id: %s" % message.message_id)
    print("correlation_id: %s" % message.correlation_id)
    key_value_pair = map_properties.get_internals()
    print("Properties: %s" % key_value_pair)
    send_callbacks += 1
    print("Total calls confirmed: %d" % send_callbacks)


def iothub_client_init():
    # 准备 iothub 客户端
    iotHubClient = IoTHubClient(connection string, protocol)
    # 设置消息过期时间
    iotHubClient.set option("messageTimeout", message timeout)
    if iotHubClient.protocol == IoTHubTransportProvider.MQTT:
        iotHubClient.set_option("logtrace", 0)
    iotHubClient.set_message_callback(
        receive_message_callback, receive_context)
    return iotHubClient


def iothub_client_sample_run():

    try:

        iotHubClient = iothub client init()

        while True:
            # 每分钟发送的信息
            print("IoTHubClient sending %d messages" % message count)

            for i in range(0, message count):
                msg txt formatted = msg txt % ((random.random() * 4 + 10),
(random.random() * 4 + 60))
                # 消息编码为 string or bytearray
                if (i & 1) == 1:
                    message = IoTHubMessage(bytearray(msg_txt_formatted,
'utf8'))
                else:
                    message = IoTHubMessage(msg_txt_formatted)
                # 可选的：assign ids
                message.message_id = "message_%d" % i
                message.correlation_id = "correlation_%d" % i
```

```
                # 可选的: assign properties
                prop_map = message.properties()
                prop_text = "PropMsg_%d" % i
                prop_map.add("Property", prop_text)
                iotHubClient.send_event_async(message,
send_confirmation_callback, i)
                print(
                    "IoTHubClient.send_event_async accepted message [%d]"
                    " for transmission to IoT Hub." %
                    i)
            # 等待命令或者退出
            print("IoTHubClient waiting for commands, press Ctrl-C to
exit")

            n = 0
            while n < 6:
                status = iotHubClient.get_send_status()
                print("Send status: %s" % status)
                time.sleep(10)
                n += 1

    except IoTHubError as e:
        print("Unexpected error %s from IoTHub" % e)
        return
    except KeyboardInterrupt:
        print("IoTHubClient sample stopped")


if __name__ == '__main__':
    print('Demo Azure IoT Hub')
    iothub_client_sample_run()
```

修改 connection_string 变量值并且替换 msg_txt 变量里的<device_id>为设备 ID。

保存脚本为 ch06_02.py。运行如下命令：

```
$ python ch06_02.py
```

程序将发送 message_count 条温度和湿度值的数据。message_count 默认大小为 3。

如图 6-27 所示是程序的一个示例输出。

可以看到用设备探测工具发送出来的消息。单击 Data 标签，然后单击 Monitor 按钮监听来自设备的消息。如图 6-28 所示是一个示例输出。

```
agusk — pi@raspberrypi: ~/Documents/book — ssh pi@192.168.0.12 — 80×28
pi@raspberrypi:~/Documents/book $ python ch06_02.py
Demo Azure IoT Hub
Info: IoT Hub SDK for C, version 1.0.13
IoTHubClient sending 3 messages
IoTHubClient.send_event_async accepted message [0] for transmission to IoT Hub.
IoTHubClient.send_event_async accepted message [1] for transmission to IoT Hub.
IoTHubClient.send_event_async accepted message [2] for transmission to IoT Hub.
IoTHubClient waiting for commands, press Ctrl-C to exit
Send status: BUSY
Confirmation[0] received for message with result = OK
    message_id: message_0
    correlation_id: correlation_0
    Properties: {'Property': 'PropMsg_0'}
    Total calls confirmed: 1
Confirmation[1] received for message with result = OK
    message_id: message_1
    correlation_id: correlation_1
    Properties: {'Property': 'PropMsg_1'}
    Total calls confirmed: 2
Confirmation[2] received for message with result = OK
    message_id: message_2
    correlation_id: correlation_2
    Properties: {'Property': 'PropMsg_2'}
    Total calls confirmed: 3
Send status: IDLE
Send status: IDLE
Send status: IDLE
Send status: IDLE
```

图 6-27

图 6-28

那么原理是什么呢？程序通过调用 iothub_client_sample_run()函数运行，然后调用

iothub_client_init()函数初始化并设置 MQTT 作为传输协议的提供者：

```
def iothub_client_init():
    # 准备 iothub 客户端
    iotHubClient = IoTHubClient(connection string, protocol)
    # 设置消息过期时间
    iotHubClient.set_option("messageTimeout", message_timeout)

    if iotHubClient.protocol == IoTHubTransportProvider.MQTT:
        iotHubClient.set_option("logtrace", 0)
    iotHubClient.set message callback(
        receive_message_callback, receive_context)
    return iotHubClient
```

然后构造要发送的消息。在这里，利用 random.random()设置温度和湿度值为随机值：

```
msg_txt_formatted = msg_txt % ((random.random() * 4 + 10), (random.
random() * 4 + 60))
# 消息被编码为 string 或者 bytearray 格式
if (i & 1) == 1:
    message = IoTHubMessage(bytearray(msg txt formatted, 'utf8'))
else:
    message = IoTHubMessage(msg_txt_formatted)
```

接着设置消息属性并通过调用 send_event_async()发送到 Azure IoT Hub：

```
# 可选的：assign ids
message.message id = "message %d" % i
message.correlation_id = "correlation_%d" % i
# 可选的：assign properties
prop map = message.properties()
prop_text = "PropMsg_%d" % i
prop_map.add("Property", prop_text)
iotHubClient.send_event_async(message, send_confirmation_callback, i)
```

程序将被执行 *n* 次。*n* 的值被定义在 message_count。

构建科学型云平台

从物理传感器设备获取完数据后，需要思考：该如何使用这些数据？在这个例子中，应该通过分析数据得到一些新的想法。机器学习、数据科学和数据挖掘都是数据分析相关的方向。

投资大量机器来计算机器学习算法是很昂贵的。一个替代的解决方案是使用一个带

有机器学习或者基于科学型的云服务。

一些云平台服务商提供数据分析服务用来方便客户管理大量的数据，其中包括各种机器学习算法。一些大公司，如微软、亚马逊和谷歌等都提供包括 IoT 板连接等很多服务。

本节将解释如何使用微软 Azure 机器学习服务。微软提供 Azure ML Studio 来开发机器学习代码，详情可以访问 https://studio.azureml.net/。

在这里的例子中，用这个服务实现分类判断一张图片中显示的是不是鸢尾花。鸢尾花的数据集在 https://archive.ics.uci.edu/ml/datasets/Iris 中可访问。

开始吧！

部署 Azure 机器学习

首先需要有一个激活的微软 Azure 账户才可以部署 Azure 机器学习。打开浏览器并访问 https://studio.azureml.net/。

Azure ML Studio 帮助用户构建机器学习模型，它提供各种机器学习的算法，包含预处理和后处理。建议阅读 Azure ML 文档：https://azure.microsoft.com/en-us/documentation/services/machine-learning/。

如图 6-29 所示展示了在 Azure ML Studio 中设计机器学习模型的示例。

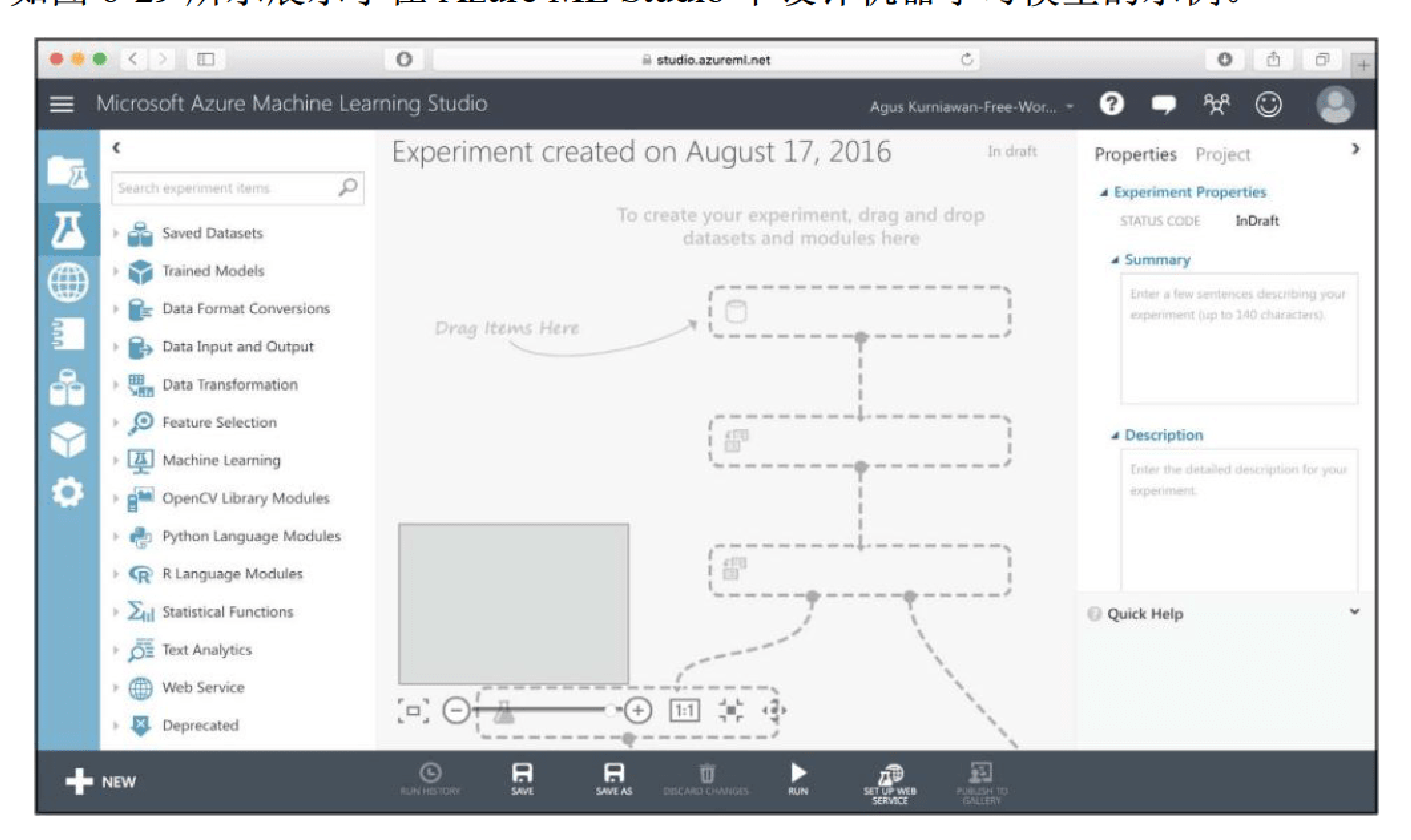


图 6-29

只需要从工具盒里通过单击和拖曳组件就可以设计模型，如使用神经网络创建一个用于分类鸢尾花的机器学习模型，如图 6-30 所示。

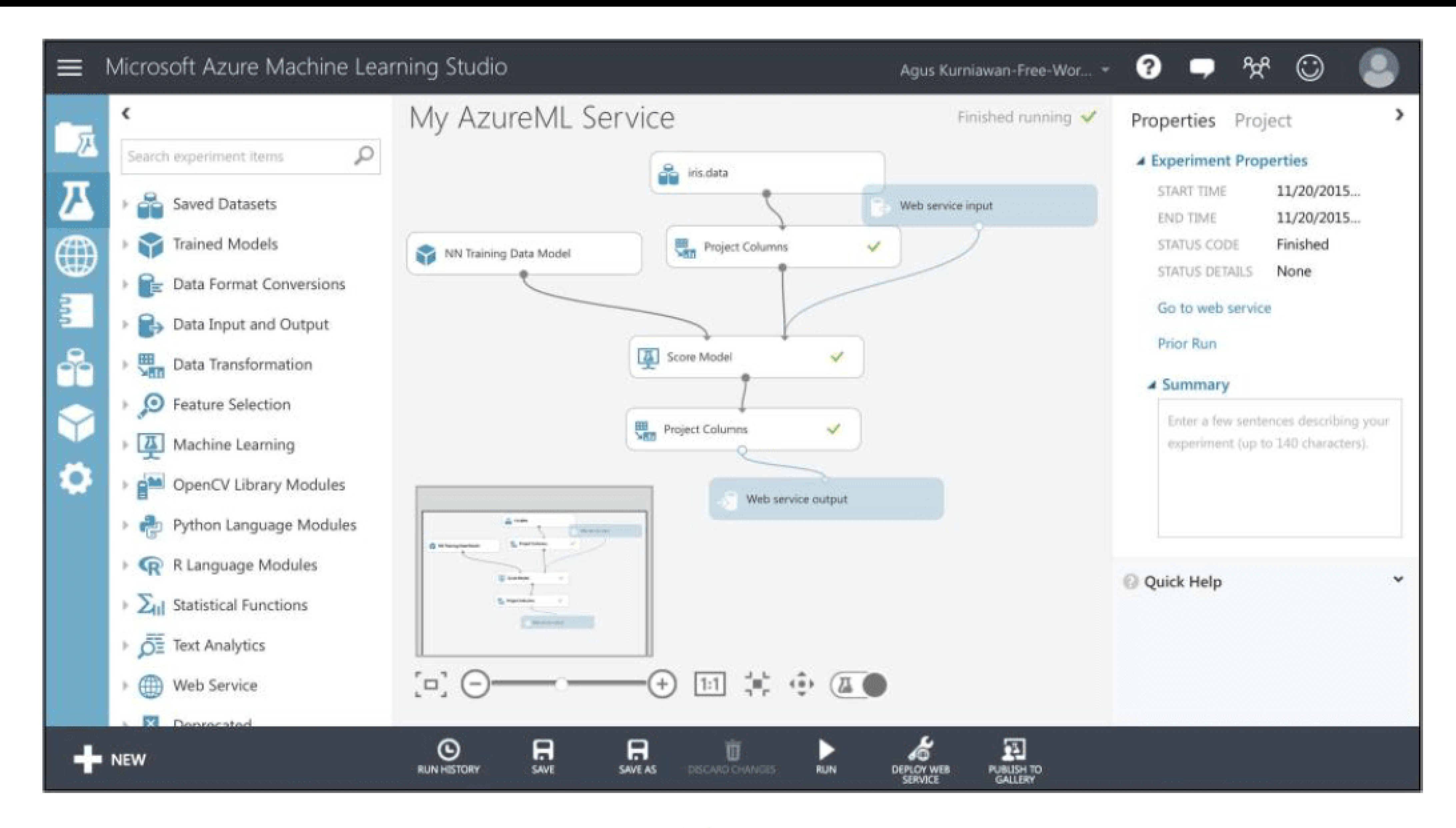


图 6-30

发布到 Azure ML 作为 Web 服务

如果已经在 Azure ML Studio 上完成构建机器学习模型，可以将它发布出去作为 Web 服务。只需要映射好 pinout 作为 Web 服务的输入和输出。之后，单击底部菜单的 DEPLOY WEB SERVICE，可以看到如图 6-31 所示界面。

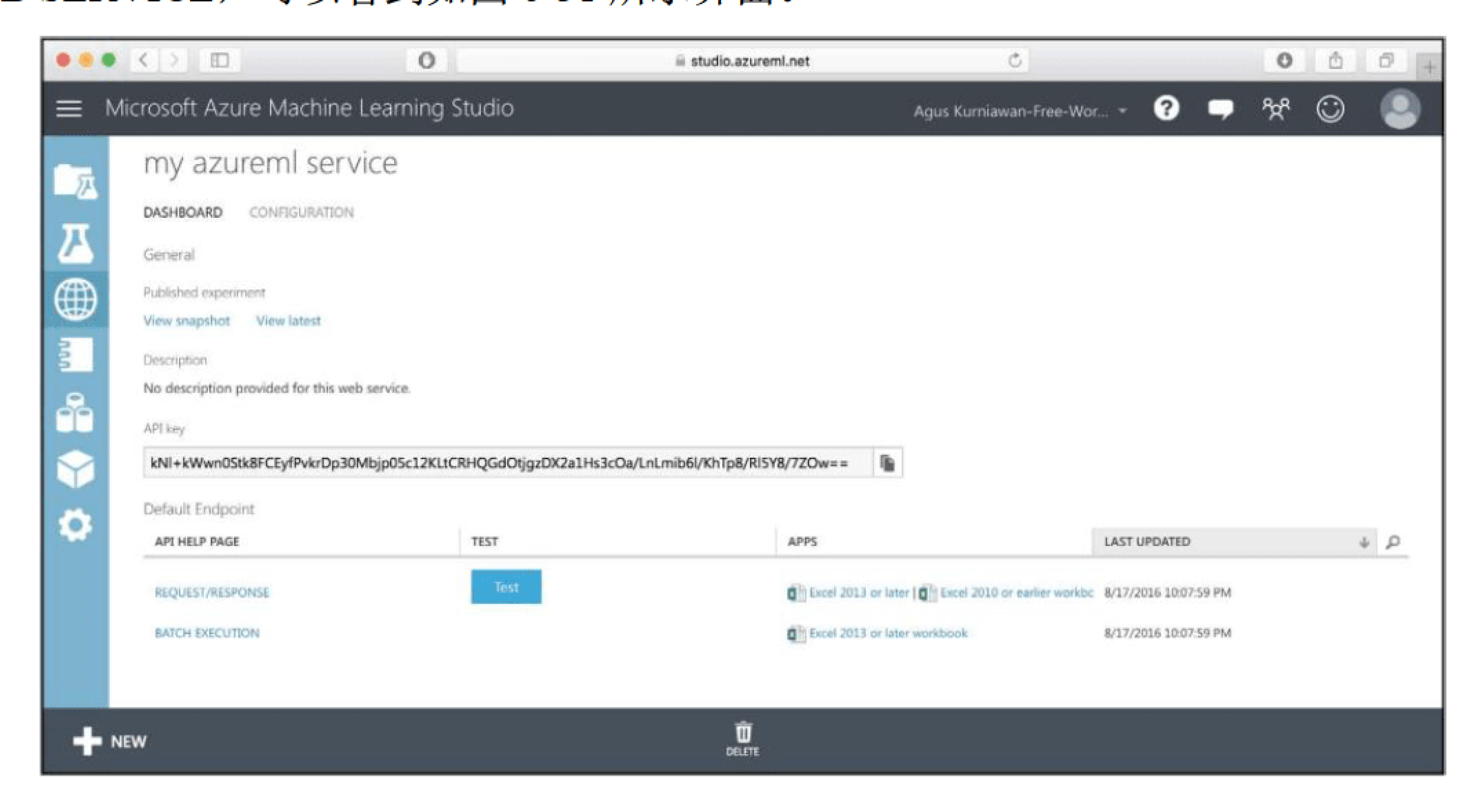


图 6-31

单击 Test 按钮进行测试，会出现一个如图 6-32 所示的对话框。

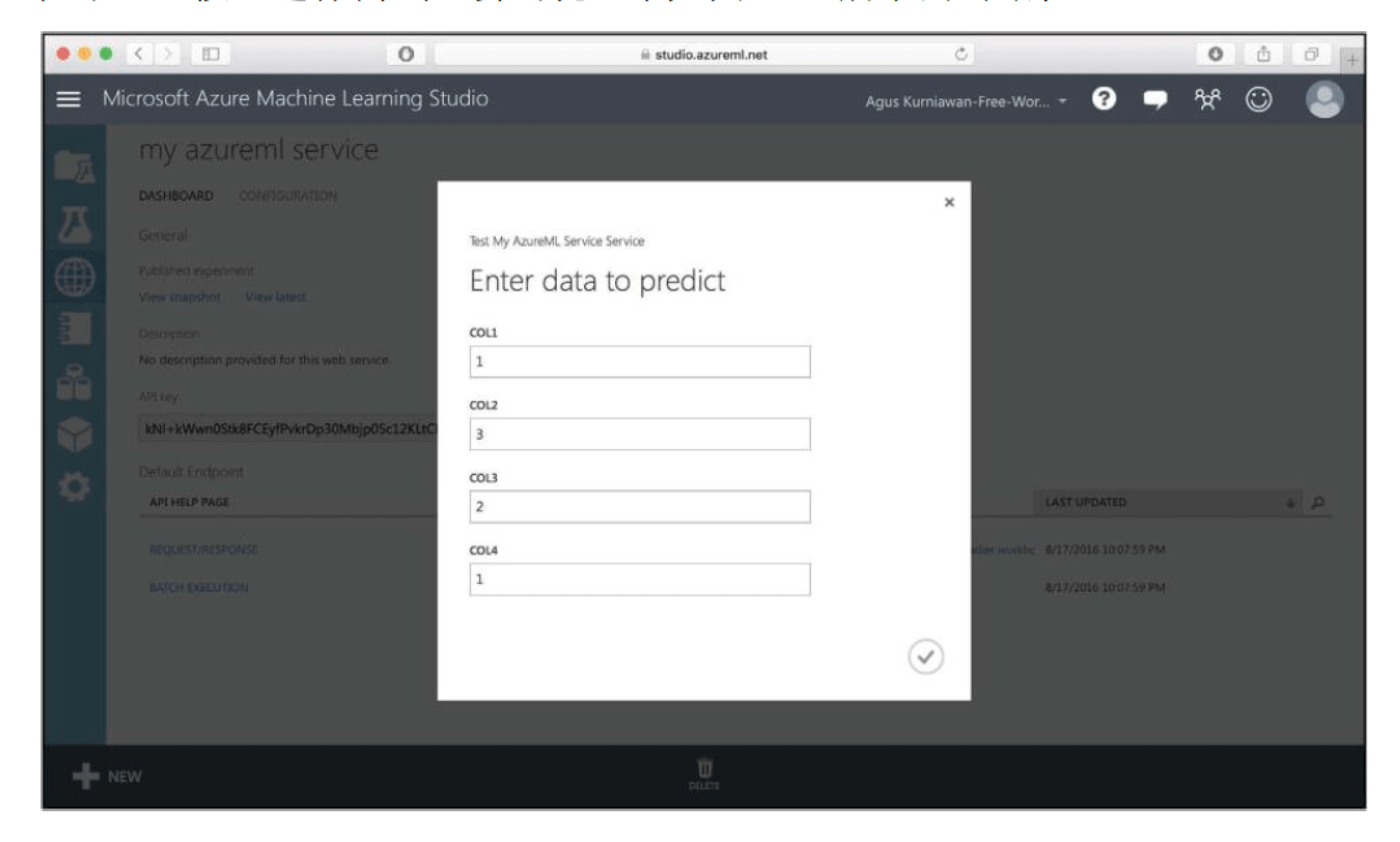


图 6-32

为 Web 服务的输入填写需要的字段。完成后，单击右下角的按钮，可以看到网站输出结果，如图 6-33 所示。

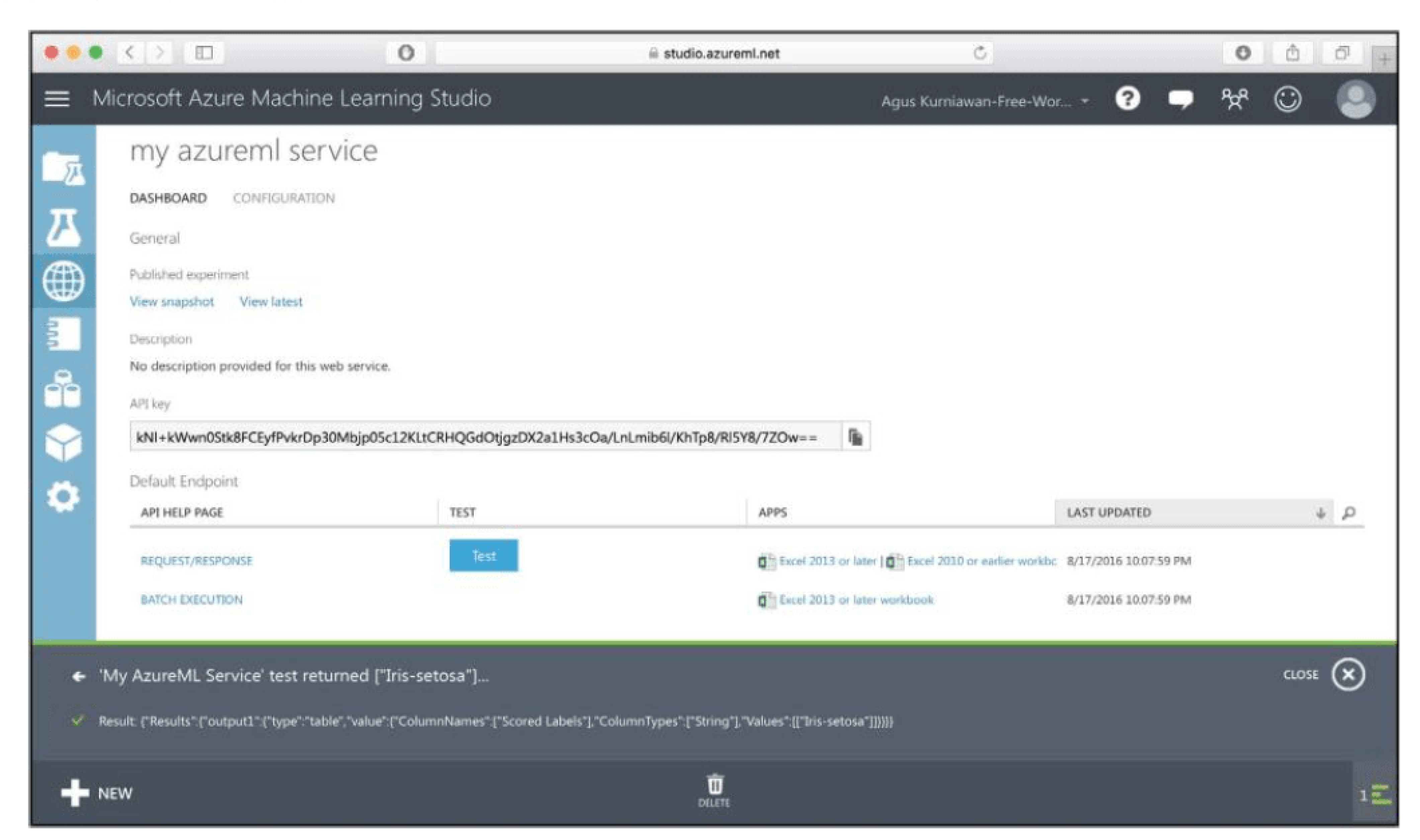


图 6-33

这个例子里，IoT 板可以被认为是 Web 服务的一部分。IoT 板用来作为传感部分，而计算部分如分类预测等都可以转移到云服务器上。

构建带有科学型数据云的 IoT 应用

在后端应用云服务对于构建一个复杂的项目是非常有帮助的，因为 IoT 板由于资源的限制，通常不能进行复杂的运算。

设想一下，假如有几个带有温度、湿度传感器的 IoT 板部署在不同的地方，它们会将感知到的数据发送到云服务器。后端的服务器由存储部分和机器学习服务器组成，后者可以计算和预测数据，如图 6-34 所示。

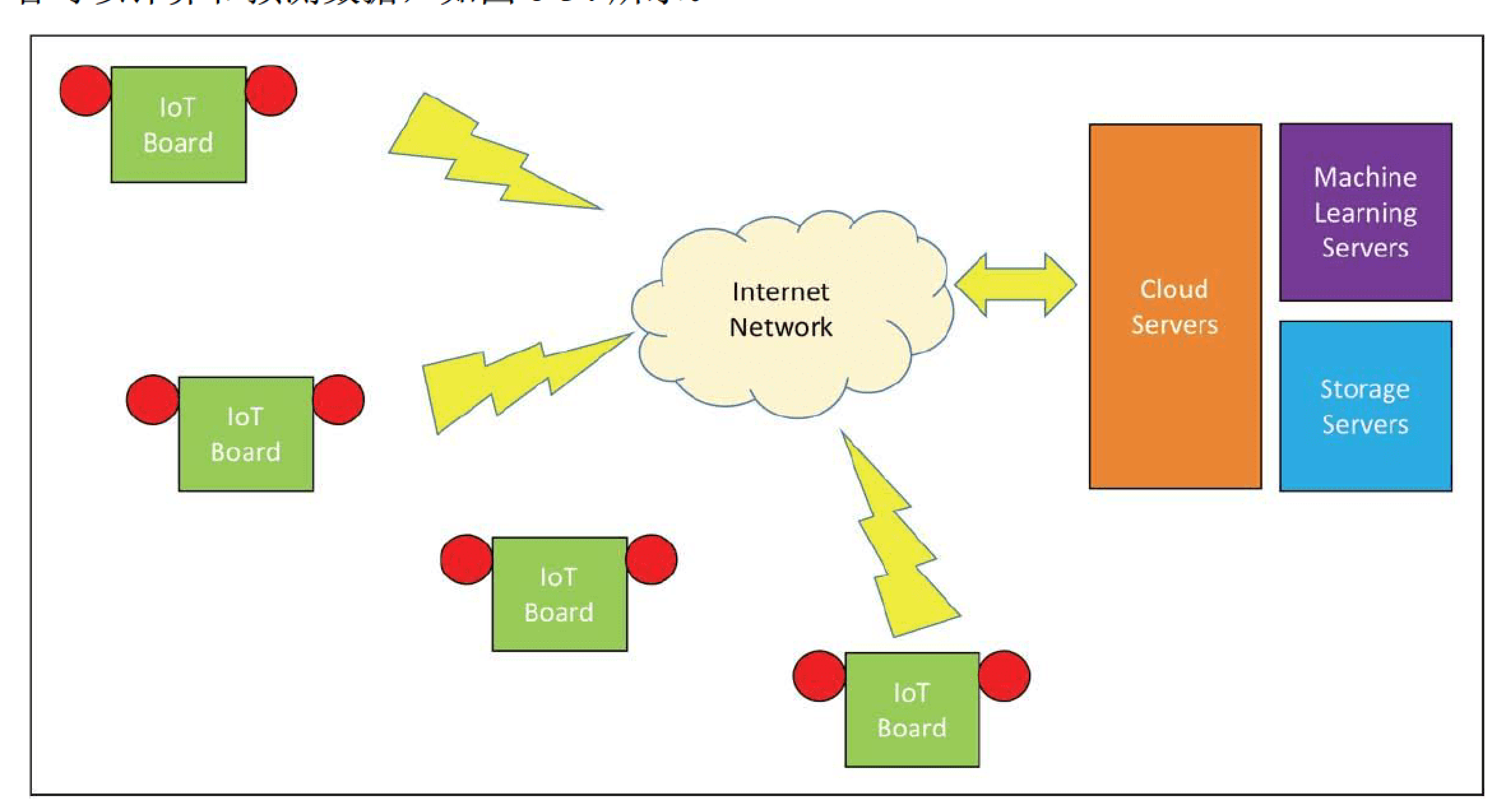


图 6-34

预测和计算的结果将会通知给用户。这只是应用的一个例子，可以将这个架构移植到自己的项目中。

总　结

在本章中，学习了一些可以被应用到 IoT 项目中的基本的云技术知识，也探索实践

了一些云平台，如 Arduino 云和微软 Microsoft Azure。我们编写了一个程序访问 Arduino 云，接着写了和微软 Azure IoT 交互的 Python 程序。最后，在 Azure 机器学习平台上构建了一个机器学习模型。

引　用

下面是一些推荐的论文、数据和网站，可以了解更多关于本章的内容。

1. Microsoft Azure IoT Hub，网址为 https://azure.microsoft.com/en-us/services/iot-hub/。

2. Microsoft Azue Machine Learning，网址为 https://azure.microsoft.com/en-us/services/machine-learning/。